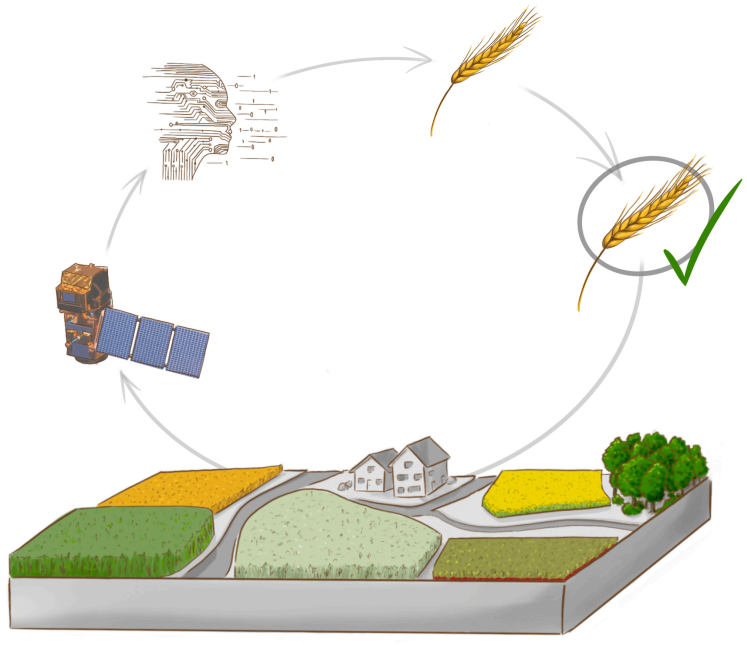
University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

# Satellite-based Crop Classification

Bachelor Thesis

Windisch, 16. August 2024




| | |
|---|---|
| Authors | Yvo Keller<br>Florin Barbisch |
| Expert | Dr. Fabian Maerki |
| Supervisor | Prof. Dr. Martin Melchior, FHNW<br>Roman Studer, FHNW<br><br>Dr. Gregor Perich, ETH Zürich |
| Client | Institute for Data Science (I4DS), FHNW |
| Project No. | 24FS_I4DS16 |

University of Applied Sciences and Arts Northwestern Switzerland, School of Engineering

## Abstract

Recognizing the need to better understand the role of pretrained geospatial foundation models in crop classification, this work addresses a key gap by fine-tuning the Prithvi model. We developed *Messis*, a model architecture built upon Prithvi. Originally pre-trained on U.S. data, Prithvi was applied to high-resolution Sentinel-2 imagery from the ZueriCrop dataset, which covers agricultural areas in Switzerland. It leverages the three-tier, remote-sensing-focused hierarchical tree structure of the labels in the ZueriCrop dataset. We created ZueriCrop 2.0, a recreation of the ZueriCrop dataset featuring 429 images with larger dimensions of 224x224 pixels, specifically to match the input requirements of *Messis* and enable its evaluation. We then assessed whether *Messis* can reduce the need for extensive labeled data that traditional machine-learning models typically require. We continued to analyze the impact of higher temporal resolution data and a new label hierarchy based on crop seasonality, comparing *Messis*'s performance to ms-ConvSTAR, a model developed by the original creators of the ZueriCrop dataset. Through 15 experiments, the properties of Messis were examined and refined, achieving a final F1 score of 34.8% across 48 crops, despite the heavily imbalanced dataset. We were able to double the performance on the F1 score when using Prithvi's pretrained weights, compared to randomly initialized weights. The results demonstrate Prithvi's adaptability to a crop classification task, handling satellite imagery from another continent with higher resolution and smaller fields. Further, we introduced a novel stratified fold-splitting method for multi-label data with counts, ensuring consistent stratification across folds.

**Keywords:**

Crop classification, Geospatial foundation models, Prithvi, Fine-tuning, Remote sensing, Earth observation, Deep learning, ZueriCrop dataset, Hierarchical classification, Time series, Stratified fold-splitting, Multi-label data with counts

## Preface and Acknowledgements

We are proud to present our bachelor thesis "Satellite-based Crop Classification", which we authored as the culmination of our Bachelor of Science in Data Science studies at FHNW. We researched, engineered, experimented, and wrote this thesis from February to August 2024.

We both have a strong passion for deep learning models, particularly foundation models like LLMs. However, this time we wanted to step away from text-based models. The limitations of running state-of-the-art models on consumer hardware, coupled with the closed-source nature of many of these models, steered us in a different direction. The fast-paced nature of the field of LLMs, driven by massive investments from large companies, also made it less exciting for us to research. Ultimately, our urge to work on something tangible and observable in everyday life is what drew us to this thesis.

The proposal for the project, and our original project agreement, included satellite-based classification of crops and cultivation methods. At the start of our work, we realized that due to model constraints, we would have to recreate the ZueriCrop dataset. Since there was no dataset available to classify cultivation methods, and creating one would require significant additional work on top, we agreed with our supervisors to limit the scope to crop classification only.

Special thanks go to our supervisors Prof. Dr. Martin Melchior and Roman Studer. They made this thesis possible in the first place by giving helpful feedback and granting us access to GPU resources. We are also extremely grateful to Prof. Gregor Perich from the crop science group at ETH Zürich for his involvement in creating the proposal for this thesis and for sharing his views, knowledge, and feedback from an agronomic perspective countless times.

We'd like to recognize Dr. Mehmet Türkoglu and Juraj Micko from ETH Zürich for providing knowledge about the ZueriCrop dataset, as well as Dr. Hamed Alemohammad from Clark University for clarifying questions about the foundation model Prithvi.

We would be remiss in not mentioning NASA and IBM for publicly releasing the foundation model this work builds upon, ESA and the European Union for funding the Sentinel-2 satellites and making the captured data available for free, and last but not least the Amt für Raumentwicklung of the Canton of Zürich and the Amt für Geoinformation of the Canton of Thurgau for providing historical data about the usage of agricultural land. Special thanks go to Sina for the beautifully illustrated cover image of our thesis. Furthermore, we'd like to acknowledge our fellow students Daniela, Thomas, and Jan for sharing their knowledge and experience of working with satellite data.

Finally, we are grateful to our families and friends for their support throughout this bachelor thesis's and our life's journey.

Yvo Keller and Florin Barbisch

Windisch, 16. August 2024

# Contents

## List of Figures

## List of Tables

# 1 Introduction

## 1.1 Motivation

Crop classification identifies and categorizes field crops using aerial and satellite imagery with AI technologies like machine learning and deep learning. It's essential and acts as a gateway technology for agricultural management, enhancing statistics, monitoring activities in real-time, and subsidy validation.

The Copernicus program of the European Union (EU) has launched six Sentinel satellites, including Sentinel-2A and Sentinel-2B, which provide valuable high-resolution, multispectral satellite imagery of the Earth at a 10x10m resolution every 5 days (ESA, n.d. b). These images are instrumental in monitoring the environment, climate, and agriculture. Automating crop classification using these images could reduce administrative burdens, streamline subsidy payment verification, minimize the need for physical inspections, improve resource management and precision agriculture, and benefit other use cases such as summary statistics (Turkoglu et al., 2021).

Existing methods for crop classification include recurrent neural network (RNN) and long short-term memory (LSTM) approaches by Rußwurm and Körner (2018), Transformer-based models by Rußwurm et al. (2020), and ms-convSTAR by Turkoglu et al. (2021), all trained and evaluated on Switzerland's comparatively small crop fields provided in the ZueriCrop dataset (Turkoglu et al., 2021). The ZueriCrop dataset provides a remote-sensing focused 3-tier tree hierarchy to combat class imbalance inherent to this dataset, which was used in the ms-convSTAR model. The ZueriCrop dataset is split into five folds by dividing the area covered by the dataset into 5 vertical bands. There is a significant drawback with this approach, as the crops are not uniformly distributed over the whole area, and therefore the resulting folds are not stratified. It is important to have comparable distributions in both the training, validation, and testing datasets and therefore folds, to ensure the model generalizes well and performs reliably across different data subsets, avoiding biases or overfitting that could arise from data discrepancies. However, stratifying multi-label datasets with counts, such as ZueriCrop, is not an easy task. As far as we are aware, no established solutions exist for this problem.

Furthermore, these traditional machine-learning and deep learning models for crop classification often require extensive labeled data and are limited in their ability to generalize across different regions or scales. Geospatial foundation models - and more generally foundation models - offer a significant advantage by leveraging vast amounts of pre-trained data, allowing them to adapt more effectively to new datasets with minimal fine-tuning (Kolides et al., 2023). These versatile, generalizable, pre-trained models are capable of performing a wide range of tasks across various domains without requiring retraining from scratch for each specific application (Jakubik et al., 2023). This adaptability stems from the models' ability to learn a broad set of features from the data during the pretraining phase, enabling them to efficiently understand and process complex information. In the context of geospatial data, foundation models are particularly valuable as they can semantically contextualize spatial data and uncover intricate patterns and relationships inherent to such datasets. Thereby, they enhance both efficiency and cost-effectiveness in geospatial analysis, and more specifically, crop classification (Stefano Ermon, 2024).

In terms of fine-tuning, these geospatial foundation models are still relatively unexplored, especially in crop classification. Jakubik et al. (2023) adapted a pretrained geospatial transformer called Prithvi for Multi-Temporal Crop Segmentation on Sentinel-2 (S2) images in the US at a 30m resolution. However, as far as we are aware, no previous work has attempted to adapt the geospatial foundation model Prithvi, originally trained on US satellite images at a 30x30m resolution, for crop classification using higher-resolution S2 imagery from Europe, specifically

Switzerland. Therefore, it is unknown how Prithvi adapts to the ZueriCrop dataset and what performance it can achieve. In particular, the influence of existing differences between the pre-training dataset and the ZueriCrop fine-tuning dataset, such as temporal resolution, different wavelengths, smaller fields, and data from a continent not seen during training, is unknown.

Our work addresses this gap in crop classification research by adapting Prithvi to handle a dataset with smaller fields and a more realistic, heavily skewed long-tail distribution of 48 crops, compared to the 13 classes used in prior US-based crop segmentation studies (Jakubik et al., 2023; Turkoglu et al., 2021). To achieve this, we leverage the ZueriCrop 2.0 dataset — an almost identical dataset to ZueriCrop by Turkoglu et al. (2021) — which includes S2 data and crop labels from Zürich and Thurgau for the year 2019. By fine-tuning the Prithvi model on the ZueriCrop dataset, we aim to advance the current state of crop classification technology.

## 1.2 Goals

The primary goal of our work is to experiment with and evaluate the foundation model Prithvi on the ZueriCrop dataset, focusing on its limits, possibilities, and obstacles during fine-tuning. We do not aim to develop a fully automated system for practical use cases, such as validation of direct payments to farmers, but rather to provide insights that could guide future developments in this area.

Our specific objectives include:

- Fine-tuning the Prithvi model on the ZueriCrop dataset and comparing its performance with existing models, such as those by Turkoglu et al. (2021).
- Investigating the impact of higher temporal resolution data and a new label hierarchy based on crop seasonality.

## 1.3 Research Questions

The research questions we will address are as follows:

1. Can the geospatial foundation model Prithvi be adapted (fine-tuned) to the ZueriCrop dataset?
   (a) What specific challenges are associated with fine-tuning Prithvi, and what methods could potentially overcome them?
2. How does the performance of the fine-tuned Prithvi model in crop classification on the ZueriCrop dataset compare to the existing models of Turkoglu et al. (2021)?
   (a) What quantitative metrics can be applied to effectively evaluate the performance differences?
   (b) Does changing the label hierarchy from a remote-sensing focused hierarchy (Turkoglu et al., 2021 / ZueriCrop) to a seasonality-based hierarchy positively impact the model performance?
3. Does the use of higher temporal resolution data have a positive effect on model performance?
   (a) What are the implications of a higher temporal resolution when using Prithvi, which was pre-trained on 3 timesteps?

## 1.4 How it was done

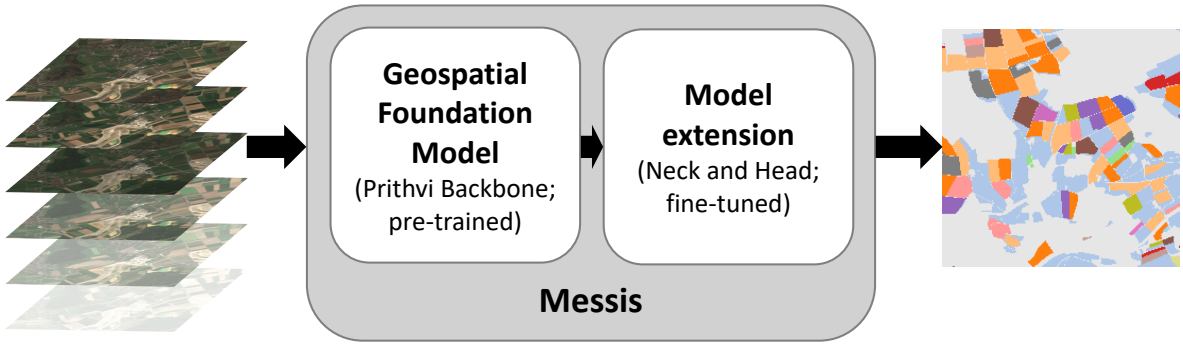



**Figure 1.1:** Our proposed model architecture Messis consists of the pre-trained geospatial foundation model Prithvi serving as a backbone in our architecture and two extensions to the foundation model: a fine-tuned neck and classification head. Messis' input consists of multiple satellite images at different timesteps and puts out the classified crop for each field.

We established a data pipeline using Data Version Control (DVC), an open-source tool for versioning, management, and reproducibility of data and machine learning models. This tool functions similarly to Git but is designed for data and models. The pipeline was used to recreate the ZueriCrop dataset with an increased chip size, which refers to the size of the images in the dataset. The new chip size is compatible with the foundation model Prithvi. The dataset was segmented into six folds to measure the uncertainty of training scores through cross-validation. We developed *Messis*, our crop classification model, and its architecture, which utilizes a frozen geospatial foundation transformer backbone, augmented with a neck to restore the image dimensions and a classification head as seen in Figure 1.1. To explore the effects of various model and data configurations, we conducted 15 experiments and synthesized those insights into a single experiment that significantly outperformed the others with an F1 score of 34.8% across 48 crops. Pixel predictions were aggregated within each field polygon, allowing us to report scores at the field majority level. We logged numerous quantitative and qualitative metrics to W&B throughout the training process and made them publicly accessible[1]. To prevent overfitting, we employed early stopping. We utilized the PyTorch Lightning trainer to minimize engineering work. The results are discussed and concluded at the end, and the code for training the model, creating the dataset, and a demo web application is publicly available on GitHub[2].

## 1.5 Contributions

We make two important contributions to the field of crop classification. First, we demonstrate how a geospatial foundation model can be adapted to a crop classification task that leverages an explicit, hierarchical tree structure of the crops and incorporates higher-resolution satellite images from a continent not seen by the model during pretraining and featuring smaller fields. Second, we propose a novel approach for splitting multi-label data with counts to ensure stratification between folds.

[1] https://wandb.ai/crop-classification/messis
[2] https://github.com/Satellite-Based-Crop-Classification/messis

## 1.6 Document Structure

This document is structured as follows:

- The Section 1 Introduction aims to give the reader an overview of why we did what and how while leaving technical details for sections further down. We also present the reader with this document structure.
- The Section 2 Related Work reviews previous studies, methodologies, applications, and challenges related to crop classification using satellite data.
- The Section 3 Data Description describes the data itself, how it was obtained and how the data pipeline prepares this data.
- The Section 4 Methodology documents the created model architecture in-depth. It further explains how this model was trained and evaluated and how the data was split into folds to report the variability of the metrics.
- The Section 5 Experiments and Results details each experiment, the results, and a rough comparison of the different setups in each experiment. It also includes Experiment 5.12 Best Model: Messis where the insights from all other experiments were combined into one.
- The Section 6 Discussion discusses the results reported in the previous section by the following aspects: brief summary, comparison to literature, interpretation, strengths/limitations, and applications.
- The Section 7 Future Work makes recommendations for future work based on our findings.
- The Section 8 Conclusion concludes the discussion and future work by concisely answering our Research Questions (RQs) and briefly summarizing our work.
- After the Section 8 Conclusion, the reader will find the References, Related Work, and Appendix.

# 2 Related Work

Crop classification is the process of identifying and categorizing field crops using data from aerial and satellite imagery (Cherlinka, 2022). It is an increasingly popular application of geospatial AI, which has emerged as a distinct field by combining geographic information systems (GIS) with AI technologies such as machine learning and deep learning. Remote sensing provides the essential data input for many geospatial AI applications. The potential applications of a model capable of classifying crops with high accuracy encompass a range of fields, including agricultural management and planning, crop rotation, and the validation of applications for subsidies (Cherlinka, 2022; Turkoglu et al., 2021).

The classification of crops is commonly influenced by three key dimensions, which are approached and handled in different ways by different methodologies: 1.) The temporal dimension helps to track crop growth over time, as the plant develops from a seedling to a fully grown plant. 2.) The spatial dimension is defined by the Ground Sampling Distance (GSD), which determines the resolution at which each pixel in the satellite imagery represents the real world. The GSD typically ranges from a coarser 30m resolution (Landsat) down to a more detailed 10m resolution (Sentinel-1/2), depending on the satellite and spectral bands. 3.) The spectral dimension involves the use of multiple spectral bands, which are each tailored to capture specific information about the land, such as vegetation health or soil moisture, based on their distinct wavelengths.

## 2.1 Applications of Satellite-based Crop Classification in the EU and Switzerland

As part of the European Space Agencies Sen4CAP program, many European countries are pushing the use of Satellite data for agricultural monitoring (Sen4CAP, 2024). While some EU countries implement cross-checking of self-declared crop data by farmers using satellite data, Switzerland relies on a well-established system where farmers manually report their crops at the beginning of each year (Schweizerische Eidgenossenschaft, 2024). Satellite-based monitoring systems have the potential of significant reductions in administrative burden for farmers (European Commission, 2024). However, there currently appears to be no immediate interest in replacing this system in Switzerland. Switzerland operates on a system of trust, with minimal field-level checks of the farmers' declarations. Governmental controlling happens on the level of the farm operation, based on plausibility checks on the written report that farmers hand in yearly (Gregor Perich, 2024b). Furthermore, in Switzerland, the direct payment system currently operates at the farm level rather than on a field-by-field basis, which limits the direct applicability of satellite-based crop classification for monitoring purposes (Gregor Perich, 2024b; Schweizerische Eidgenossenschaft, 2024). Despite this, certain individual crops and ecological practices, such as high-stem trees, rapeseed, and the creation of biodiversity areas, are eligible for direct payments and could potentially be monitored using satellite data (BLW, 2024). However, these payments currently also depend on additional conditions that cannot be captured by crop classification systems alone, such as specific labor investments by the farmer. As a result, while satellite data could play a role in monitoring and validation, the overall utility of crop classification systems in Switzerland remains limited due to the broader, farm-level focus of the direct payment system.

In contrast to this, EU countries have direct applications for crop classification systems today. **Poland** has implemented an advanced S2 based crop classification system to enhance agricultural statistics. The system, part of the SATMIROL[3] project, combines satellite data with existing administrative and cadastral information to automatically detect and monitor crop types across the country. Preliminary results from 2020 showed an overall classification accuracy of 79%, with high accuracy for major crops like winter wheat but challenges for minority

[3] https://stat.gov.pl/en/experimental-statistics/gospostrateg/satmirol/

classes (Statistics Poland, 2024). **Hungary** has developed a system for verifying agricultural subsidy claims, utilizing S2 satellite imagery to monitor sowing and ploughing events at the field level. The system, which is part of a pilot project, combines random forest algorithms for crop classification with NDVI (Normalized Difference Vegetation Index) time series analysis to detect cultivation events. It achieves an overall accuracy of 88% for 22 vegetation classes (Henits et al., 2022). In **France**, a new pilot system is being tested where farmers receive direct payments based on the actual crops planted, with verification primarily through Sentinel satellite data. This system, part of the future CAP 2023/2027[4] program, aims to simplify the payment verification process by using satellite imagery to automate eligibility checks and reduce on-site inspections. Farmers are notified via the Telepac[5] platform, which provides real-time status updates on their fields — green for compliance, orange for pending review, and red for discrepancies. The goal is to streamline the subsidy process and potentially eliminate the need for physical inspections if satellite data suffices (Agence de services et de paiement Francaise, 2024). In **Denmark**, the Agricultural Agency uses Sentinel satellite imagery to monitor fields and detect activities such as mowing or plowing in real time. The system developed by company DHI[6] provides farmers with live updates on their fields' status through a web-GIS portal, which facilitates compliance checks without physical inspections. Time series analysis of Sentinel-1 (S1) and S2 satellite data enables the detection of agricultural events and reduces the need for on-site visits (Eskesen and Nyborg, 2021). Denmark has a traffic light map system in place that is similar to France. If a field is categorized as red, the direct payment is rejected and farmers need to object if they think not rightly so. That way, when the model classifies wrongly, the farmers can file an objection by taking geo-referenced photos of their affected plot using a provided mobile app. One mentioned expected advantage of this traffic light map is the effect of nudging farmers with fields that do not meet the requirements to withdraw their application (Eskesen and Nyborg, 2021).

This shows that neighboring EU countries are more advanced than Switzerland in the use of satellite technology. Crop classification serves as a gateway technology, enabling a range of further satellite-based applications (Perich, 2023). By identifying the locations of specific crops, it becomes possible to develop yield models and offer tailored recommendations for field management, such as optimized use of pesticides and fertilizers (Perich, 2023). This capability can pave the way for various practical use cases, from precision agriculture to improved resource management. The technology for crop classification has been proven to be effective for agricultural monitoring in neighboring countries, demonstrating its potential to significantly augment agricultural statistics. Now, it's an opportune time for policymakers to consider implementing this technology and to reap the administrative benefits that it offers. Although the systems introduced in the EU are not without flaws, if satellite-based crop classification proves to be effective and resilient, Switzerland might see similar advancements in its agricultural practices over time (Perich, 2023).

## 2.2 Model Architectures used in Crop Classification

Researchers have used different types of model architectures to tackle the challenge of crop classification, ranging from traditional RNN approaches to the more recent Transformer architecture.

[4] https://agriculture.ec.europa.eu/common-agricultural-policy/cap-overview/cap-2023-27_en
[5] https://www.telepac.agriculture.gouv.fr
[6] https://eo.dhigroup.com/projects/monitoring_agricultural_fields/

### 2.2.1 RNN/LSTM approaches

As an example of RNN/LSTM approaches in crop classification, Rußwurm and Körner (2018) used a temporal sequence of S2 images of a large area in Germany to develop a phenological model for vegetation classification. By visualizing internal activations over a series of cloudy and non-cloudy images, they discovered that some recurrent cells lower the input activity for cloudy observations. They inferred from this that the model inherently learns to filter clouds. This allowed them to achieve state-of-the-art accuracy in 2018 on a large number of crop classes while performing minimal preprocessing. Their proposed model architecture is a bidirectional sequential encoder, incorporating convolutional RNN and LSTM cell variants to encode the temporal sequence both in sequential and reversed order into a fixed-length representation. The concatenated final cell states are then passed to a convolutional layer for classification.

### 2.2.2 Transformer Architecture

The advent of transformer models has revolutionized various domains within AI, originally natural language processing (NLP) and, more recently, vision tasks. Introduced in the paper "Attention is All You Need" by Vaswani et al. (2023), transformers fundamentally altered how sequential data is processed by replacing recurrent architectures in favor of self-attention mechanisms. The transformer architecture allowed for greater parallelization during training and significantly improved performance on many NLP tasks. Vision Transformers (ViTs), as proposed by Dosovitskiy et al. (2021), adapted the transformer architecture to image data by treating images as sequences of patches. This approach demonstrated that transformers could achieve state-of-the-art results on image classification tasks, rivaling traditional convolutional neural networks (CNNs).

The rapid adoption of transformer models has for a big part been made possible by the consistently increasing availability of computing power. Advances in hardware, such as the development of powerful graphics processing units (GPUs), and the optimization of distributed computing techniques have made it possible to train large-scale transformer models (Toews, 2023).

Rußwurm et al. (2020) employed an attention-based Transformer model for crop classification in their study and found that it slightly outperformed recurrent models, such as LSTM (Rußwurm and Körner, 2018) and StarRNN (Turkoglu et al., 2019). These recurrent models generally showed better performance than Random Forests and convolution-based models, with the exception of the TempCNN model (Pelletier et al., 2019).

### 2.2.3 ConvSTAR

In a recent paper from ETH Zürich, "Crop mapping from image time series: Deep learning with multiscale label hierarchies", Turkoglu et al. (2021) presented the ConvSTAR model and the ZueriCrop dataset. ConvSTAR employs convolutional recurrent neural networks (ConvRNNs), which replace the standard matrix multiplications in RNNs with convolution operations to better capture spatial dependencies. According to their work, the new cell structure called STAR makes ConvSTAR particularly efficient, requiring fewer parameters than traditional LSTM or GRU cells (Turkoglu et al., 2021).

The hierarchical ConvSTAR network (ms-ConvSTAR) is designed with multiple stages, each consisting of two ConvSTAR layers. The output from each stage is passed to the next, creating a deep, layered representation of the data. The different network stages predict an incrementally more granular label resolution in the hierarchical tree. On top of that, they apply a label

refinement network to encourage consistency between predictions at the different stages. This refinement network takes the initial predictions for each tier and processes them further to produce a final, more accurate prediction for the finest label granularity (Turkoglu et al., 2021).

### 2.2.4 Prithvi

Foundational models such as large language models (LLMs) are pre-trained unsupervised on large amounts of data, using a task like next-token prediction, and thus gain some level of general understanding of the data they are trained on. This is what allows a model such as GPT, which is based on a decoder-only transformer architecture, to generate text conditioned on an input prompt. Such foundational models can then be fine-tuned on more specific tasks, which can significantly enhance their performance in that specific area, while requiring much less training data than a model trained from scratch would require (Jakubik et al., 2023).

IBM Research together with NASA brought this approach, previously primarily used in the area of NLP, to the realm of Geospatial artificial intelligence. Their work "Foundation Models for Generalist Geospatial Artificial Intelligence" introduced Prithvi, a transformer-based geospatial foundation model (Jakubik et al., 2023). It is pre-trained on more than 1TB of multispectral satellite images of the US, sourced from the Harmonized Landsat-Sentinel 2 (HLS) dataset. This is achieved through the use of an encoder-decoder structure, where the encoder learns to produce embedding representations of the satellite images in the latent space, which the decoder then utilizes to reconstruct masked-out sections of the input images. The decoder module of Prithvi can be discarded and replaced with a neck module whose purpose is to restore the input dimensions. This is followed by a classification head, with the encoder now serving as the backbone of this new model architecture. They fine-tuned such models on various downstream tasks like flood mapping, wildfire scar mapping, multi-temporal cloud gap imputation, and multi-temporal crop segmentation. When compared to training the model for these tasks using randomly initialized weights, they find that Prithvi's pre-trained weights accelerates the fine-tuning process, while performing well against state-of-the-art on these tasks. Moreover, the amount of data utilized in this process can be significantly reduced without compromising the model's accuracy, showcasing data efficiency (Jakubik et al., 2023). The model parameters for Prithvi are open-sourced through Huggingface[7].

In the context of our research, Prithvi plays an essential role. We will conduct a series of experiments to adapt the foundation model on the ZueriCrop data set. Our objective is to determine the potential performance of this model. Another important aspect is the comparison to the ConvSTAR model by Turkoglu et al. (2021), which has been trained on the same dataset.

## 2.3 Common Challenges

Different challenges add to the difficulty of building crop classification models. Understanding them is essential to making the right decisions that will lead to a model that performs well in the real world.

### 2.3.1 Cloud Coverage

Satellite imagery is often obstructed by cloud cover, posing a challenge for remote sensing applications. Rußwurm and Körner (2018) and Turkoglu et al. (2021) have shown that the performance of RNN models remains relatively consistent regardless of the extent of cloud coverage in the input data. Notably, the study by Turkoglu et al. (2021) incorporated the use of

[7] https://github.com/NASA-IMPACT/hls-foundation-os

cloud masks during training and achieved similar performance metrics to those obtained with less cloud-affected data. This suggests that RNNs are capable of effectively learning to ignore or compensate for cloud-covered areas in satellite images. It is important to note that this approach is effective when applied to a multitude of timesteps, as exemplified by the 72-timestep approach utilized by Turkoglu et al. (2021). However, it is probable that the outcome would be less optimal when the number of timesteps is constrained. An alternative approach for such cases is to set a maximum threshold for cloud coverage, based on S2 metadata, directly when choosing the input data. This ensures the model obtains a reasonable amount of usable input data for its classification task.

### 2.3.2 Class Imbalance

Another common challenge in crop classification, and generally supervised classification, is the issue of class imbalance. This arises because certain crop classes have a higher occurrence as they are more commonly farmed than others due to higher demand. An example for this is Winter Wheat (Gregor Perich, 2024b). Additionally, regional differences can further complicate this issue, as variations in climate, soil quality, and agricultural practices across different areas lead to uneven distribution of crop types. In many practical applications, unusual classes are just as important as or even more so than frequent ones. This also applies to crop classification, where crops of high ecological or economic value (biodiversity areas, vegetables, and orchards, for example) are significantly less common than wheat fields or meadows (Turkoglu et al., 2021). In conventional training regimes, machine learning models tend to ignore infrequent and underrepresented classes. They prioritize the dominant classes instead to optimize the cumulative performance across the entire dataset (Dong et al., 2018).

Turkoglu et al. (2021) try to combat class imbalance by introducing a hierarchical classification approach, in which they break down coarse classes into more specific ones. For example, a pixel can be labeled as "Field crops" in Tier 1, then "Small Grain Cereal" in Tier 2, and most granular as "Summer Wheat" in Tier 3. The class hierarchy was defined by domain experts in consideration of the main types of land use in Swiss agriculture and the visual characteristics of the crops in satellite imagery. As part of their work, Turkoglu et al. (2021) published the "ZueriCrop" dataset, which covers an area of 50 km $\times$ 48 km in the Swiss cantons of Zurich and Thurgau. The dataset consists of a greater number of classes (48) with a much less balanced class distribution compared to other existing datasets, thus making it more representative of the real world.

### 2.3.3 Ground Truth Quality

Although the quantity of data available for agricultural monitoring using Sentinel satellites has significantly increased in recent years, largely due to numerous projects across Europe under the Sen4CAP initiative by ESA, not much is known about the quality of this data (Sen4CAP, 2024). Additionally, a recent report based on a postulate in the Swiss National Council highlighted numerous areas for improvement in agricultural data management at both the federal and cantonal levels. For instance, it was noted that there are currently five different systems used across the Swiss cantons for agricultural data collection, which indicates a generally fragmented data landscape at the administrative level, that makes the data management system prone to errors and inconsistencies (Swiss Parliament, 2022). According to Gregor Perich (2024b), there may be instances where the provided labels do not accurately reflect the ground truth. There are several reasons for this:

- In Switzerland, during January and February of each year, farmers record their planned harvest for each field they farm. However, this is subject to change, as environmental

reasons such as too much rain may lead the farmer to switch to a crop more suited to those environmental conditions. (Gregor Perich, 2024b)

- Human error in data collection is another source of mislabelling, as the polygons are drawn by hand from aerial photographs, and the crops are also entered manually (Gregor Perich, 2024b).
- The government only makes plausibility checks of logical nature (e.g. if the drawn bounds match with property boundaries) (Gregor Perich, 2024b).
- Some fields may grow two types of crop consecutively within a year, while only the main crop is recorded by the government in the data collection process (Gregor Perich, 2024b). In this case, there is a risk that a satellite image taken on a particular day may not correspond to the crop type.

Depending on how often incorrect labels are present, they can introduce significant noise into the training process, degrading performance, especially for classes that are already underrepresented.

# 3 Data Description

## 3.1 Satellite Data

We originally intended to use the publicly available ZueriCrop dataset[8] by (Turkoglu et al., 2021). The dataset contains 28x28 pixel chips, but the transformer backbone (Prithvi) of our model Messis requires images of the dimension 224x224 pixels. The dataset did not include any spatial information, so merging the chips and creating fewer but bigger chips was impossible. Thus, we needed to recreate the dataset from scratch by downloading the ground truth labels and satellite data. We name this dataset ZueriCrop 2.0 and refer to it as such throughout this thesis.

In our case, the features used for crop prediction are derived from satellite data. Multiple data products are available for this purpose. Notably, NASA has two satellites: LANDSAT 8 and 9, which revisit each point on the earth in an 8-day cycle. Both satellites are sun-synchronous, meaning they visit the equator at the same time during each orbit (Landsat Science Outreach Team, 2021). This ensures consistency and is critical in assessing time-series data. The satellites acquire images with a 30m x 30m per pixel resolution and cover a 180km-wide swath, which is the width of the area captured in a single pass of the satellite over the Earth's surface, at a 15° angle (Gross et al., 2022).

The European Space Agency (ESA) operates two satellites as part of their S2 mission, Sentinel-2A and Sentinel-2B. These satellites have nearly identical Mean Local Solar Time (MLST), and their spectral bands' central wavelengths are virtually equal for Near Infrared (NIR), Red-Green-Blue (RGB), and Short-Wave Infrared (SWIR 1/2). They have a 20.6° field of view corresponding to an image swath of approximately 290 km. They fly in the same orbit, but are phased at 180°, such that one satellite is on the opposite side of the earth from the other satellite, resulting in a revisit cycle of 5 days. S2's resolution is much better at 10x10m per pixel for the RGB and NIR bands and a 20x20m resolution for the SWIR 1/2 bands (ESA, n.d. b).

This led to the creation of the "Harmonized Landsat and Sentinel-2" (HLS) project. This initiative from NASA aims to create a combined record from the Instruments aboard Landsat-8/9 and Sentinel-2A/B (NASA, 2022). This dataset is also what Prithvi was pre-trained on, as explained in Section 2.2.4 Prithvi. The addition of S2 data to Landsat 8/9 proved to increase the performance practically the same as just using the S2 data in multiple use cases (Astola et al., 2019; Forkuor et al., 2018; Paul & Kumar, 2019). The S2 red-edge bands and the better spatial resolution are explaining factors according to Astola et al. (2019), which renders HSL useless for these applications. We still see applications for HLS where a high visiting frequency is important. As Prithvi restricts us to only 3 timesteps, this does not apply to our use case and we used only satellite imagery from S2.

## 3.2 Agriculture Data

The ground truth labels are based on the "Landwirtschaftliche Kulturflächen" (Cultivated agricultural areas) defined by Bundesamt für Landestopografie (2022), which each canton must provide yearly (Schweizerische Eidgenossenschaft, 2024). This data is used to implement the Agriculture Act, particularly as a basis for calculating direct payments. Additionally, the Federal Statistics Act necessitates the spatial recording and exchange of agricultural holding and area data throughout Switzerland (Bundesamt für Landwirtschaft BLW, 2022).

As Section 2.3.3 Ground Truth Quality mentions, the farmers record the crops they intend to cultivate between the 15th of January and 28th of February of each year (Schweizerische

[8] https://github.com/0zgur0/multi-stage-convSTAR-network

Eidgenossenschaft, 2024). However, the farmers might deviate from their original plans and plant alternative crops due to environmental factors such as rain (Gregor Perich, 2024b). Farmers might also plant catch crops during the year, but only the main crop type is recorded in the data. Consequently, the data does not always represent what happened in the fields.

Since there is no centralized source for downloading merged historical data for the whole of Switzerland, the data for 2019 (the same year as used for the original ZueriCrop dataset) was individually acquired from the cantons of Thurgau and Zürich. To obtain a dataset comparable to ZueriCrop by Turkoglu et al. (2021), individual emails were sent to the "Departement für Inneres und Volkswirtschaft (DIV) - Amt für Geoinformation" of the canton of Thrugau and the "Amt für Raumentwicklung - Abteilung Geoinformation" of the canton of Zürich.

The dataset consists of geo-referenced polygons representing fields or areas, each mapped to an LNF ("Landwirtschaftliche Nutzfläche") code that specifies the crop grown there (e.g., apples, wheat, meadow, etc.). These LNF codes were not systematically created; instead, they grew historically based on administrative needs. A detailed reference for the different LNF Codes and their meanings can be found in appendix Table A.4. To make more sense of these codes, Turkoglu et al. (2021) reduced them to 48 classes. The resulting label hierarchy is described in the following Section 3.2.1 Label Hierarchy, and is what we generally refer to when mentioning the label hierarchy.

### 3.2.1 Label Hierarchy

The label hierarchy of the ZueriCrop and ZueriCrop 2.0 dataset breaks down the 48 classes into three tiers (Turkoglu et al., 2021). The distribution of the classes in each tier is shown in Figure 3.1. A more in-depth look at the hierarchy and how classes between tiers are related is presented in Figure 3.2. It can be easily observed that within each subset of classes sharing the same parent class, the distribution of instances follows a long-tail pattern. A detailed hierarchy table including LNF Codes can be found in Appenix A.1 ZueriCrop 2.0 Dataset with ZueriCrop Label Hierarchy.

- The first level of the class hierarchy was designed with two main purposes: (i) to categorize the primary types of land use found in Swiss agriculture, and (ii) to group crop types based on their visual characteristics as seen in satellite imagery. For example, field crops, which are planted in rows, can be easily identified by remote sensing technology. Grasslands, which form the largest class in the dataset, are a diverse group that includes various grass types, mixtures, and other land use forms. These are characterized by their persistent greenery, high biomass, and generally short plant height. Other first-level classes include orchards, special crops, and forests. Orchards can be recognized by their distinctive planting patterns. Special crops include a wide variety of less common crops such as asparagus, various berries, and herbs, each with its own set of subclasses.
- The second level of the hierarchy provides more detailed subdivisions of the first-level classes, reflecting plant families and cultivation practices. For instance, small grain cereals, which are cultivated in closely spaced rows with a uniform planting density and an upright canopy structure, are grouped together. In contrast, the broadleaf row crops class consists of dicotyledonous plants that are also grown in rows, but have a horizontal leaf orientation, unlike grain crops.
- At the third level, the hierarchy differentiates between individual crop species, offering the highest level of detail. However, in cases where specific species information was not available, second-level labels were repeated instead. For the forest class, the same label is used across all three levels of the hierarchy.


**Figure 3.1:** The ZueriCrop 2.0 dataset class distribution over the three tiers. Logarithmic scale.

**ZueriCrop Label Hierarchy** (Bands represent number of pixels in dataset)

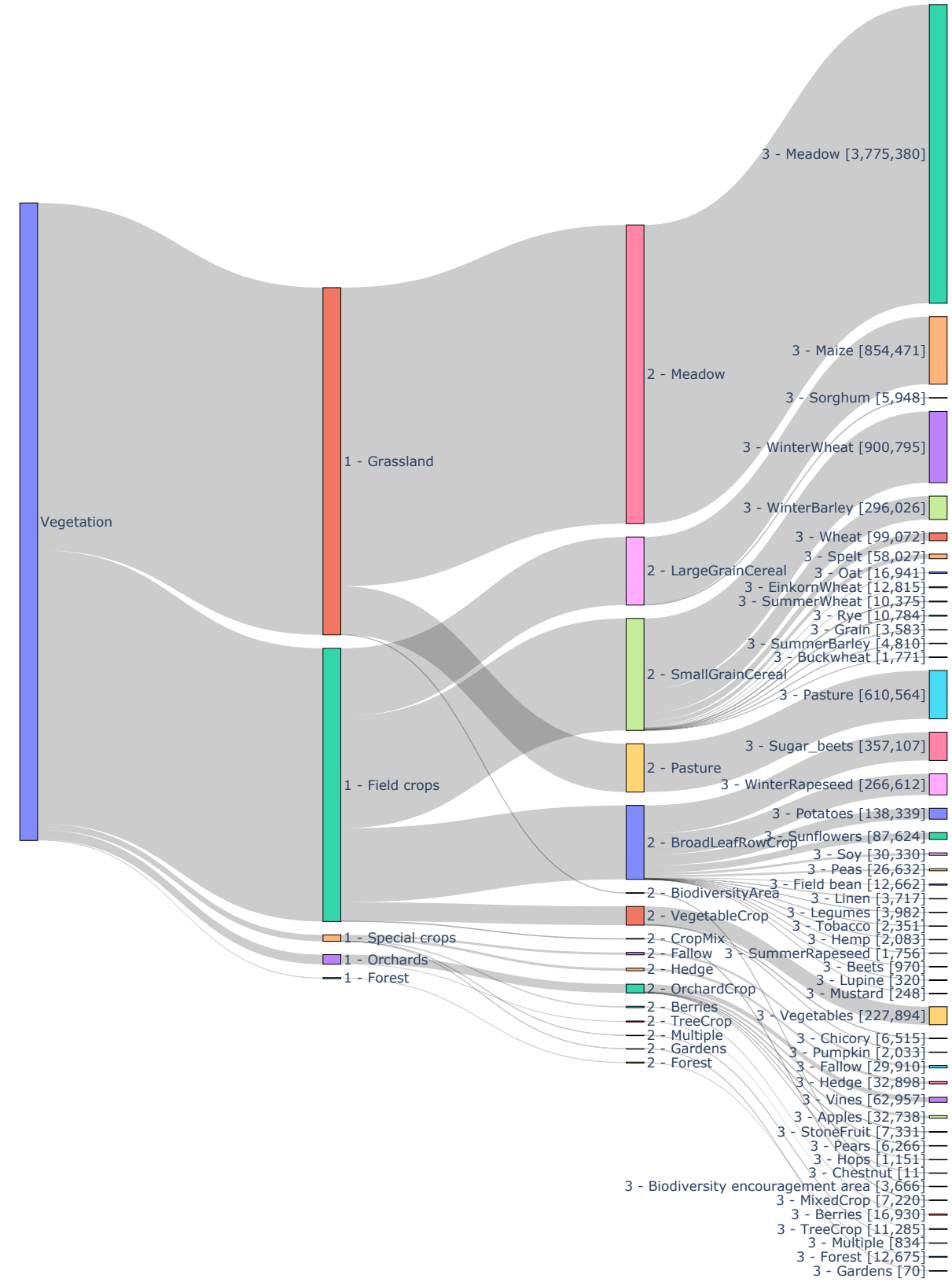



**Figure 3.2:** Sankey diagram of the ZueriCrop 2.0 dataset in the ZueriCrop Label Hierarchy.

## 3.3 ZueriCrop Dataset

In this section, we summarize the differences between the original ZueriCrop dataset and ZueriCrop 2.0 (our reproduction of ZueriCrop). Since both datasets are very similar and we exclusively use ZueriCrop 2.0 in our work, we will not provide an in-depth description of the original ZueriCrop dataset.

The main difference between the two datasets are:

- The biggest difference is the size of the image chips the dataset is split into. The ZueriCrop dataset uses 28x28 chips, while the ZueriCrop 2.0 dataset is composed of 224x224 chips.
- The ZueriCrop dataset is composed of 142 timesteps (but the authors only used 71 to train ms-ConvSTAR) while the ZueriCrop 2.0 dataset comprises only 3 timesteps.
- The ZueriCrop 2.0 dataset uses the 6 channels that were used to pre-train Prithvi, while the ZueriCrop dataset contains 9 channels (Turkoglu et al., 2021). The bands B05 (705 nm), B6 (740 nm), and B7 (783 nm) were dropped.
- Another difference is the bounding box covering the area. ZueriCrop covers the area (8.364, 47.240, 9.000, 47.697) and the ZueriCrop 2.0 dataset covers a slightly larger area, resulting in a more convenient image size of 5120x5120: (8.364, 47.240, 9.0405, 47.69894).

## 3.4 Our ZueriCrop 2.0 Dataset

As Prithvi only accepts 3 timesteps, we do not benefit from the higher temporal resolution in the HLS product. Therefore, we chose to use only S2 data, as it also provides a higher resolution. We utilized the same bands that were chosen from the HLS product to pre-train Prithvi (Jakubik et al., 2023). These 6 S2 bands are listed in Table 3.1, along with their use cases. We downloaded the data preprocessed on the most advanced Level-2A (refer to Figure 3.3 for the different processing levels), and only for the area covering our dataset (8.364, 47.240, 9.0405, 47.69894). This saved on bandwidth and disk space, as we could leverage Cloud Optimized GeoTIFFs (COGs) from the "Sentinel-2 Level-2A" collection, provided by "Earth Search by Element 84"[9]. The bounding box was inspired by Turkoglu et al. (2021), but widened to an image size of 5120x5120 pixels. This was a better fit to the chip size of 256x256 pixels, which we originally intended to work with.

[9] https://stacindex.org/catalogs/earth-search#/43bjKKcJQfxYaT1ir3Ep6uENfjEoQrjkzhd2?t=3

| Name | No. | Wavelength (Bandwidth) | Resolution | Use cases |
|---|---|---|---|---|
| Blue | B2 | 490nm (65nm) | 10m | Soil/vegetation discrimination, forest type mapping, identifying man-made features, illuminates material in shadows, penetrates clear water |
| Green | B3 | 560nm (35nm) | 10m | Contrast between clear and turbid water, highlights oil on water surfaces, vegetation, man-made features visible |
| Red | B4 | 665nm (30nm) | 10m | Identifying vegetation types, soils, urban areas, limited water penetration |
| NIR | B8 | 842nm (115nm) | 10m | Mapping shorelines, biomass content, detecting and analyzing vegetation |
| SWIR1 | B11 | 1610nm (90nm) | 20m | Measuring moisture content of soil and vegetation, contrast between vegetation types, differentiate between snow and clouds |
| SWIR2 | B12 | 2190nm (180nm) | 20m | Measuring moisture content of soil and vegetation, contrast between vegetation types, differentiate between snow and clouds |

**Table 3.1:** S2 spectral bands used in the pertaining of Prithvi and later fine-tune Messis and their use-cases according to Sinergise (2023). Note that S2 has more than these 6 spectral bands.

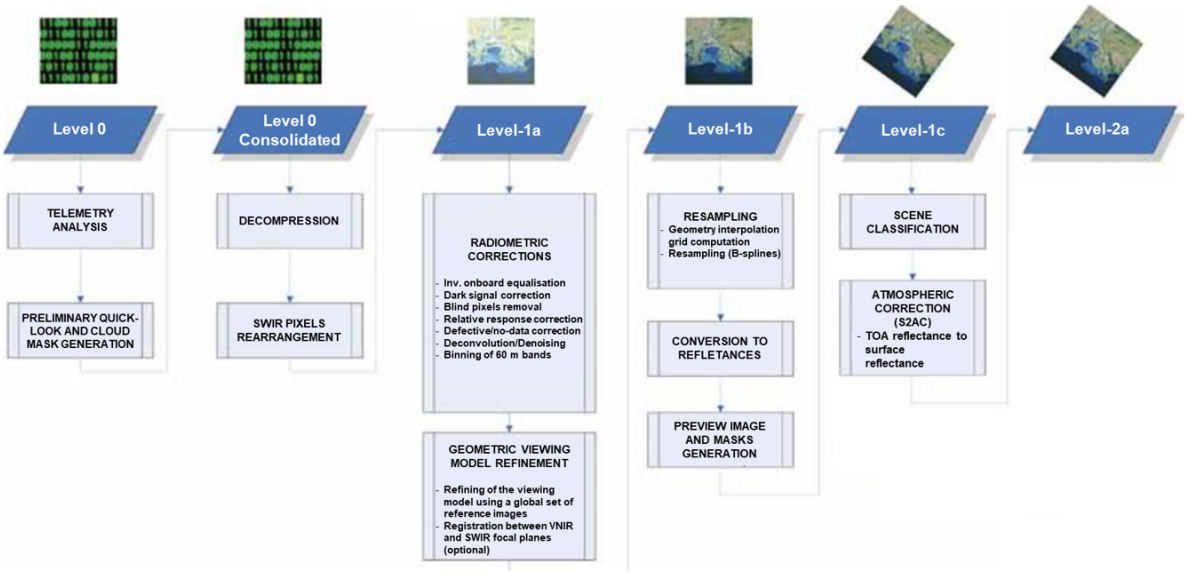



**Figure 3.3:** Processing levels from Level-0 to Level-2A (ESA, n.d. a)

As our model can only process 3 timesteps, we needed to reduce the number of timesteps accordingly. This was done by selecting images with less than 5% cloud cover that were captured between March 1, 2019, and September 30, 2019. Then, the first and last image in this date range was selected, together with the image closest to the midpoint between the dates of the first and last image. The chosen images were from the following dates: 21.03.2019, 26.06.2019, and 29.09.2024. The time difference between the first and second image is 97 days, and 95 days between the second and third image. Experiment 5.3 Satellite Imagery Time Ranges evaluates

different time ranges and Experiment 5.11 More Timesteps evaluates the usage of more than three timesteps.

The polygons were analyzed for overlaps into the other cantons. Polygons lying outside the canton from which the data was acquired were discarded due to overlaps with polygons from the other canton, and to reduce the number of chips with minimal ground truth. The remaining polygons were subsequently rasterized into an image of the same size and area as the satellite data. A pixel was only considered to be inside a polygon if its center lay within the polygon, rather than just its borders touching the polygon. Three GeoTIFF images, embedded with georeferencing information covering the considered area, were created with the following properties as the value: a) the LNF code of each polygon, b) the number for each category in the three-tier label hierarchy of the ZueriCrop dataset, and c) a unique field ID.

Then, the images (ground truth and satellite data) were split into 224x224 pixel chips. Chips with no ground truth data were discarded. These chips were subsequently assigned to 6 folds. The strategy we utilized to split the resulting chips into 6 folds for cross-validation in our experiments is our own approach, minimizing the mean kullback-leibler divergence. It is described in Section 4.6 Cross-Validation and Data Split Strategies. Alongside our strategy, we describe 3 more strategies in that section, which are compared to each other in Experiment 5.1 Dataset Fold Split Strategy.

The resulting chips are normalized to have a mean of 0 and a standard deviation of 1 for each channel. This normalization is based on the means and standard deviations of the training folds. So in the case of a 5-fold cross-validation, we recompute the means and standard deviations for each fold-run using only the 4 folds used for training.

The input data was further augmented with flips and jitter for Experiment 5.5 Improving Generalization with Data Augmentation and Experiment 5.12 Best Model: Messis. The images were randomly flipped horizontally and/or vertically according to a hyperparameter $flip\_prob$, which controls both the probability of a vertical and a horizontal flip. The same flip was applied to both the image and the target mask. Subsequently, normally distributed noise with a mean of 0 and standard deviation controlled by another hyperparameter was added to the image.

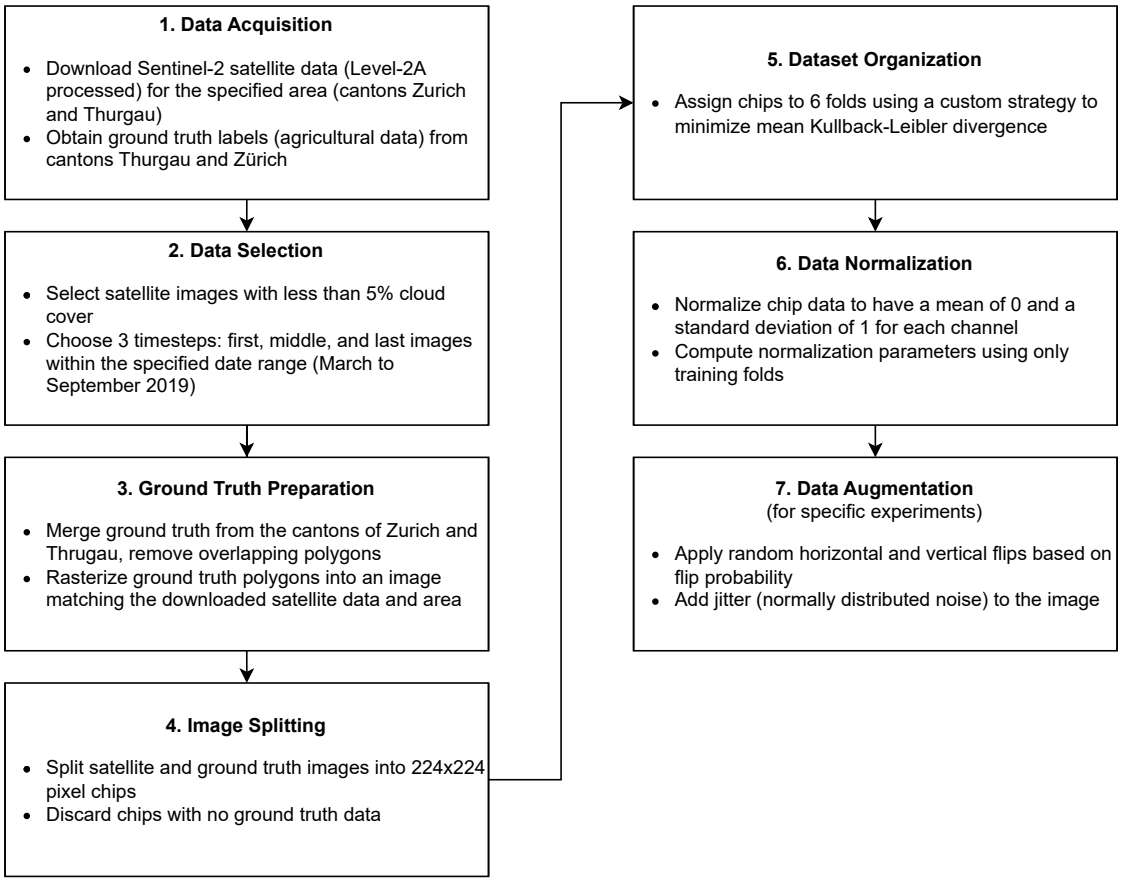



**Figure 3.4:** Data processing pipeline for creating the ZueriCrop 2.0 dataset.

This data pipeline described above and visually shown in Figure 3.4, was implemented using DVC[10] (Data Version Control). DVC is a data versioning tool that keeps track of the input and output files of each stage in the pipeline, as well as the parameters used (e.g. the covering area, number of timesteps, time range, etc.). DVC furthermore creates a dependency graph and monitors changes to dependencies, such as previous stages, parameters, and input files. With a single command, it can regenerate the affected data, adjusted to the new settings. This ensures that the data used for training always matches the parameters in version control, thus ensuring consistent results and reproducibility of training runs. Additionally, this allows for quick iteration of changes to the training dataset, based on simple parameter changes.

[10] https://dvc.org/

# 4 Methodology

## 4.1 Model Architecture

There are number of different factors to consider when making the choice for an appropriate model architecture. This section will go over the architecture of our model Messis[11], and the reasoning behind the design choices that were made. At the highest level of abstraction, Messis is composed of three modules that can be seen in Figure 4.1. The first one is the backbone, Prithvi, a Vision Transformer encoder pre-trained on geospatial data. Followed by that, the neck, forming the bridge between the token-based output of Prithvi and the classification head. Finally, the decision-making component in the form of the hierarchical head, responsible for interpreting the processed features and classifying the crops. A detailed model architecture graph including input/output shapes of each layer is shown in the appendix on Figure A.5.

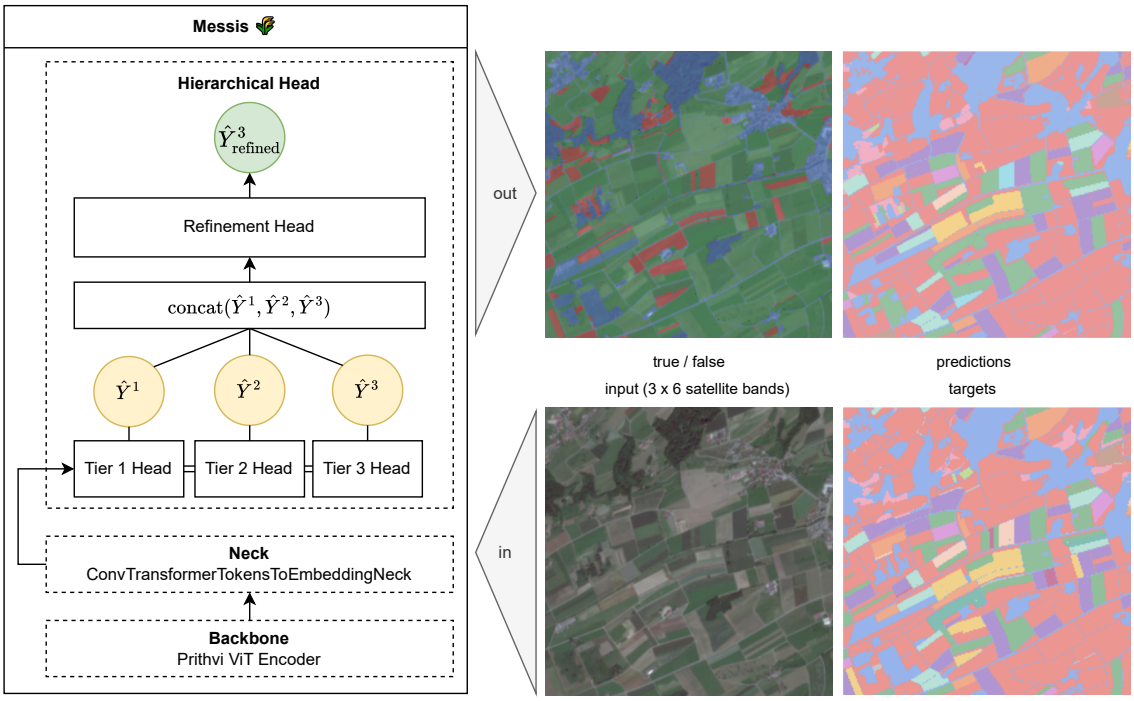



**Figure 4.1:** Messis Model Architecture. Blue indicates background. Top right image: predicted fields, colored by class identified. Bottom right image: target fields, colored by class.

### 4.1.1 Backbone: Prithvi Vision Transformer

The original transformer is built to process a sequence of 1D token embedding vectors. In contrast to a language transformer, in a ViT one token does not represent a (sub)-word, but instead an image patch of dimension $P \times P \times C$, where $C$ represents the spectral dimension (number of channels). The image patches are created by splitting an input image of dimension $H \times W \times C$ into $N$ non-overlapping patches. Each patch is a block from the original image, and the number of patches $N$ is given as $N = \frac{H \times W}{P^2}$. Each patch block is then flattened and projected into constant embedding vector dimension $D$ with a trainable linear projection. Positional information is provided to the ViT by adding positional encodings to the resulting patch embeddings (Dosovitskiy et al., 2021).

[11] Messis is the latin word for Harvest

So far, we have looked at how the spatial and spectral dimension of an image is handled in the ViT. The temporal dimension in our dataset is given by the multiple timesteps of satellite imagery that show crop growth over time. The paper "ViViT: A Video Vision Transformer" introduced the concept of a tubelet embedding, where non-overlapping spatio-temporal tubes are extracted from the input volume (Arnab et al., 2021). The number of timesteps $T$ considered for a tube is the tubelet size ($TS$), resulting in the dimension $T \times H \times W \times C$ for a tube. This approach is what is used in Prithvi to extend the original Vision Transformer (ViT) architecture by a temporal dimension.

In the ViT, a learnable embedding is prepended to the sequence of patch embeddings. At the Transformer encoder output, this learnable embedding becomes a representation of the image which we passed as input. This is analogous to BERT's `[class]` token, which is trained to represent the (masked) input sentence and serves as the primary feature for downstream tasks (Devlin et al., 2019).

Transformers lack the translation equivariance and locality properties that are inherent to CNNs by their design. Consequently, they require a larger training dataset to generalize well. Experiments with large-scale training demonstrated that inductive bias (Dosovitskiy et al., 2021) could be outdone. Large-scale training is provided with Prithvi, which was pre-trained using the Masked Autoencoder (MAE) approach on over 1TB of multispectral satellite imagery (Jakubik et al., 2023). MAE is a self-supervised learning approach, where part of the input data is masked. The encoder processes the visible parts of the data, and the decoder is then trained to reconstruct the missing parts from the encoder's latent representations (K. He et al., 2021). It is important to note that the MAE approach is what enables effective training on geospatial data, such as satellite imagery. Unlike other domains, we often lack extensive labeled data for satellite images. The MAE framework, with its self-supervised learning approach, allows the model to learn meaningful representations of the geospatial data by reconstructing masked parts of the input in a self-supervised fashion.

Messis relies on the encoder component of Prithvi as its backbone. Prithvi itself has a requirement: The input image size must be 224x224, so that it can be split into 14x14 patches of 16x16 ($P = 16$) pixels[12]. We considered this input dimension in the creation of our ZueriCrop 2.0 dataset, to make sure the classification can be done on a per-pixel basis.

The Prithvi encoder component starts with a patch embedding layer that projects the satellite imagery of dimension $BS \times C \times T \times H \times W = 1 \times 6 \times 3 \times 224 \times 224$ through a 3D convolutional layer into an embedding space of dimension $D = 768$. For demonstration purposes, we'll assume a batch size of 1 ($BS = 1$) in all input/output tensor dimensions that follow.

The convolutional layer uses a kernel size and stride of $(TS, P, P) = (1, 16, 16)$, effectively separating the input into non-overlapping patches. This operation results in a tensor of dimension $1 \times 768 \times 3 \times 14 \times 14$, where 768 is the embedding dimension $D$, 3 represents the number of timesteps $T$ and $14 \times 14$ results from dividing the spatial dimensions $224 \times 224$ by the patch dimension $16 \times 16$.

We can express the output tensor dimensions as:

$$\text{Output Tensor Shape} = \left(D, \frac{T}{TS}, \frac{H}{P}, \frac{W}{P}\right) \tag{4.1}$$

[12]Contrary to LLMs, the sequence length in most Vision Transformer architectures is typically limited due to technical and computational constraints. This is due to the fact that ViTs use fixed positional encodings that are precomputed based on the maximum sequence length (number of patches). These help the model learn and maintain the spatial structure of images.

Each element in this tensor is a $D$-dimensional embedding representing a specific patch from one timestep of the input. The subsequent step flattens and transposes this tensor to dimensions $1 \times 588 \times 768$. Here, 588 is the total number of patches across all timesteps ($3 \times 14 \times 14 = 588$), and 768 is the embedding dimension. This operation produces patch embeddings for all $N = 588$ patches.

To these patch embeddings, their positional encoding is added. In Prithvi, a fixed positional encoding is created by extending the 2D sine-cosine positional encoding from the original MAE to a 3D version. This is done by creating 1D encodings for height, width, and time dimensions and then combining them into a 3D positional encoding that captures dimensions $T \times H \times W$ (Jakubik et al., 2023). A zero-vector of dimension $1 \times D$ is prepended to account for the `[class]` token. The resulting patch embeddings are then passed through all 12 sequential transformer blocks that follow.

### 4.1.2 Neck: From Tokens to a Single Spatially-Structured Embedding

The neck component of Messis is an important bridge between the token-based output of the Prithvi ViT and the subsequent layers of the model. Its main function is to transform the sequence of token embeddings into a single, spatially-structured embedding suitable for the downstream crop classification task. Here, we continue to closely follow the implementation of the model for Multi-Temporal Crop Segmentation by Fraccaro et al. (2023) which was one of the multiple applications of Prithvi in the research of Jakubik et al. (2023).

The input to the neck is a tensor of shape $1 \times 589 \times 768$ produced by the transformer encoder, where 589 represents the number of tokens (588 patch tokens plus one `[class]` token), and 768 is the embedding dimension. The first step in the neck's operation is the removal of the class token, reducing the number of tokens to 588. Contrary to use cases where we are interested in a global representation of the entire input, in our crop classification task, the `[class]` token becomes redundant and potentially counterproductive. We want to make fine-grained pixel-specific predictions, rather than a single classification for the whole input.

These 588 tokens are then reshaped into a 3D grid of $3 \times 14 \times 14$, corresponding to the three timesteps and the 14x14 patch grid of the original input. This reshaping process preserves the spatio-temporal relationships encoded by the Vision Transformer.

The core of the neck's functionality is its use of transposed convolutions (also known as de-convolutions) to upsample the token grid. The neck applies four sequential ConvTranspose2d operations, each with a kernel size of 2 and a stride of 2. These operations are organized into two stages, each containing two transposed convolutions with a Norm2d and GELU non-linearity inbetween.

$$\begin{aligned} \text{Output Size} = &(\text{Input Size} - 1) \times \text{Stride} \\ &- 2 \times \text{Padding} \\ &+ \text{Dilation} \times (\text{Kernel Size} - 1) \\ &+ \text{Output Padding} + 1 \end{aligned} \tag{4.2}$$

Using equation (4.2), we can calculate that each transposed convolution doubles the spatial dimensions of its input. After four such operations, the 14x14 grid thus is expanded to 224x224, matching the dimensions of the original input image.

Simultaneously with the spatial upsampling, the neck increases the channel dimension from 768 to 2304. This increase is achieved through the first transposed convolution, which uses 2304

kernels, effectively tripling the channel dimension to account for the three timesteps. The transposed convolutions that follow maintain this channel dimension while continuing to upsample spatially.

The final output of the neck is a tensor of shape $1 \times 2304 \times 224 \times 224$, where 2304 represents the concatenation of the three timesteps along the channel dimension ($3 \times 768 = 2304$). This output retains the temporal information from the input. At the same time, it provides a spatially-structured representation that aligns with the original image dimensions, such that we can now start classifying crops.

### 4.1.3 Head: Spatially-Structured Embedding to Crop Classes

Finally, the head component of Messis is what allows the model to do what it was developed for in the first place: predict a crop class for each pixel in the satellite imagery. At this point, it becomes important to consider the crop class hierarchy that was established in our ZueriCrop 2.0 dataset. The goal is to build a model that learns the hierarchy that consists of $T$ tiers, and can make predictions at a more granular level with each subsequent tier $t$.

The head component of Messis starts with as many hierarchical heads in sequence as there are tiers in the label hierarchy. Each head is comprised of two sequential convolutional modules, where a module is defined as a Conv2d layer with 3x3 kernel size, followed by BatchNorm2d and a ReLU activation function. The first head reduces the number of channels from the original 2304 down to 256, all further heads keep this channel dimension. Following the ConvModules, we apply Dropout2d. This configuration enables the ConvModules to transform the features such that the final Conv2d layer of the head, which projects the 256 channels down to match the number of classes $N_t$ in that tier, can identify the crop class. This convolutional layer uses kernel size 1x1. Each hierarchical head returns the logits (assigning a probability for each class to each pixel), and the features transformed by the ConvModules. Thus, the logits are now in dimension $1 \times N_t \times 224 \times 224$, and the features in dimension $1 \times 256 \times 224 \times 224$. The logits are kept aside, while the transformed features are passed forward as input to the hierarchical head for the next tier, where they are processed the same way.

When predicting a hierarchy, consistency is essential. There is nothing that forces the model to strictly adhere to the hierarchy. It is free to perform predictions for different tiers that are in conflict with one another. To illustrate, a pixel might be assigned the prediction *Grassland* in tier 1 and *Vegetable Crop* in tier 2. This violates the parent-child relationship of the hierarchy. To encourage the model to favour coherent labels across the hierarchy levels, we employ a refinement module, similar to the one employed in the ms-ConvSTAR crop classification model (Turkoglu et al., 2021). This module takes the concatenated logits of all prior hierarchical heads as input, and produces a final prediction for the last, most granular tier, while capturing the relationships between them. It consists of a series of convolutional layers, including 1x1 and 3x3 convolutions, batch normalization, ReLU activations, and Dropout1D for regularization. The result of this final module is the final prediction for each pixel in the given satellite image, in dimension $1 \times N_T \times 224 \times 224$.

## 4.2 Evaluation Approach

The label hierarchy necessitates certain considerations in our evaluation approach. Given the available data on fields in the satellite imagery, it makes sense to differentiate between evaluating on pixel level and field level. Additionally, predictions that adhere strictly and consistently to the given label hierarchy necessitate the derivation of class predictions for higher Tiers 1 and 2 from Tier 3.

### 4.2.1 Pixel-Wise vs. Field Majority Evaluation

There are two approaches used to infer the predicted class for a field in different tiers. The rasterization of the field polygon results in a field mask, which indicates which pixels on the input satellite image are part of a specific field.

#### Pixel-Wise

In the pixel-wise method, each pixel within the field mask is evaluated individually. Using softmax activation, the model's raw outputs (logits) are converted to probabilities. The class with the maximum probability is chosen as the predicted class for that pixel. The performance score is calculated by comparing the predicted class of each pixel to its target class. This makes this approach sensitive to fine-grained details, as each pixel contributes to the overall score.

The predicted class for each pixel is given by:

$$\hat{y}_{ijk} = \arg\max_{c} P_{ijk}(c) \tag{4.3}$$

where:

- $P_{ijk}$ is the probability that pixel $(i, j)$ in image $k$ belongs to class $c$.
- $\hat{y}_{ijk}$ is the predicted class for pixel $(i, j)$ in image $k$.

#### Field Majority

In the case of field majority, instead of evaluating each pixel individually, a majority voting approach is utilized. The model outputs probabilities for each pixel, similar to the pixel-wise method. However, instead of selecting the highest probability class for each pixel, the predicted classes of all pixels within a field mask are aggregated to determine the most common class. This is done by applying softmax onto the logits, using argmax to determine the predicted class for each pixel, and then designating the class that was most frequently predicted within a specific field as the majority class. The score is calculated such that each pixel within that field accounts for a prediction of that majority class. While the background class is considered in the pixel-wise method, it is ignored in the field majority approach. This field-level evaluation is less sensitive to noise and small errors in individual pixel predictions.

The predicted class for each pixel in the field when using the field majority method is thus given by:

$$\hat{y}_f = \arg\max_{c} \sum_{(i,j)\in M_f} \mathbf{1}(\hat{y}_{ij} = c) \tag{4.4}$$

where:

- $\mathbf{1}(\cdot)$ is the indicator function, which is 1 if the condition inside is true and 0 otherwise.
- $\hat{y}_{ij}$ is the predicted class for pixel $(i, j)$.
- $M_f$ is the set of pixels belonging to field $f$.
- $\hat{y}_f$ is the majority predicted class for field $f$.
- $c$ is each possible class label.

### 4.2.2 Deriving Classes from the Label Hierarchy

A key decision that was made for computing the metrics, is the way we derive the predicted classes from the label hierarchy. As Messis has multiple heads, each of which predicts the class on a specific tier, one possible approach would be to take the outputs of the Tier 1 head, Tier 2 head as well as Tier 3 refinement head, and apply the pixel-wise and field majority methods on their output logits. However, as outlined in Section 4.1.3 Head: Spatially-Structured Embedding to Crop Classes, this approach does not guarantee consistency in the parent-child relationships of the label hierarchy.

Instead, the pixel-wise and field majority evaluation methods are applied solely to the output logits of the Tier 3 refinement head, and for metric computation the higher-tier classes are derived from their parent-child relationships within the label hierarchy. This guarantees consistency, as for example if *Maize* is the predicted class for a pixel in Tier 3, this pixel will always be accounted for as *Large Grain Cereal* in Tier 2 and *Field Crops* in Tier 1.

While this consistency-guaranteeing approach has its benefits, we want to acknowledge a potential limitation. It can be illustrated with a specific scenario. Imagine a field where the Tier 3 refinement head predicts, on each pixel, a 30% probability for *Wheat*, 30% for *Oat*, and 40% for *Vines*. In Tier 2, Vines map to *Orchard Crops*, while Wheat and Oat map to *Small Grain Crops*. Using field majority or pixel-wise methods directly on Tier 3 outputs, the pixels would be classified as Vines, and thus as an Orchard Crop in Tier 2. However, if we first aggregated the probabilities from Tier 3 based on their mapping to the Tier 2 classes and then applied field majority or pixel-wise methods, we would classify the pixel as Small Grain Crop, which could potentially be more accurate given the combined higher probability for Wheat and Oat of 70%. In essence, this approach does overlook the cumulative probabilities across related classes in lower tiers, possibly leading to less accurate higher-tier predictions.

## 4.3 Tracked Metrics

In order to gain a comprehensive understanding of the model performance, we employ a combination of weighted and macro-averaged metrics, as well as confusion matrices and segmentation masks. The focus is on those that are most fundamental in evaluating Messis in the context of potential use cases. However, the choice of metrics is crucial not only for accurately measuring model performance, but also for ensuring comparability with benchmarks such as the ConvSTAR model by Turkoglu et al. (2021), which to our knowledge is current state-of-the-art on the heavily imbalanced ZueriCrop dataset with 48 classes (Turkoglu et al., 2021). Messis is trained on a reconstructed version of the ZueriCrop dataset (see Section 3.3 ZueriCrop Dataset) and uses the same label hierarchy. As such, the ability to make comparisons with the metrics achieved in that study is essential to our research. It helps us determine whether a pre-trained geospatial foundation model like Prithvi provides any advantage over training from scratch.

The metrics that will be introduced in this section are calculated and logged for each phase, mode, and tier. The phase is either the training or validation phase, and for the final model, the test phase. For mode, we differentiate pixel-wise and field majority calculation modes, as described in Section 4.2.1 Pixel-Wise vs. Field Majority Evaluation. The number of tiers is dependent on the label hierarchy. For the ZueriCrop label hierarchy, there is Tier 1, Tier 2, and Tier 3, as can be seen in Section 3.2.1 Label Hierarchy. We also discuss the rationale behind their selection, and how they address the challenges posed by our dataset.

### 4.3.1 Weighted Accuracy

Accuracy is one of the most commonly used metrics for classification problems. It indicates the percentage of correctly classified samples in relation to the total number of samples. An advantage of accuracy is its simplicity and interpretability.

#### Overall Accuracy

The formula for overall accuracy is given by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

where:

- $TP$ (True Positives) are the number of correctly predicted positive samples.
- $TN$ (True Negatives) are the number of correctly predicted negative samples.
- $FP$ (False Positives) are the number of incorrectly predicted positive samples.
- $FN$ (False Negatives) are the number of incorrectly predicted negative samples.

However, evaluating model performance only on overall accuracy is misleading. It can mask poor performance on minority classes, because it is dominated by the majority classes. In our crop classification dataset, a few classes represent the majority of the samples, while many others are underrepresented. This imbalance poses significant challenges, as models tend to be biased towards the majority classes, leading to high accuracy for these classes, but poor performance on the minority classes (Dong et al., 2018). Therefore, alternative metrics are necessary to provide a more complete picture of the model's performance across all classes, including those that are underrepresented.

#### Weighted Accuracy

One of the metrics we employ for the evaluation of our model is weighted accuracy. Weighted accuracy accounts for this imbalance by assigning different weights to each class based on their frequency in the dataset. This means that the performance on more frequent classes has a greater impact on the overall metric. A good result in this metric thus indicates that our model is able to predict majority classes well, which is a valuable practical outcome.

Weighted accuracy is calculated as follows:

$$\text{Weighted Accuracy} = \sum_{i=1}^{N} w_i \cdot \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{4.6}$$

where:

- $N$ is the total number of classes.
- $w_i$ is the weight for class $i$, typically proportional to the number of samples in class $i$.
- $TP_i$ (True Positives) are the number of correctly predicted positive samples for class $i$.
- $TN_i$ (True Negatives) are the number of correctly predicted negative samples for class $i$.
- $FP_i$ (False Positives) are the number of incorrectly predicted positive samples for class $i$.
- $FN_i$ (False Negatives) are the number of incorrectly predicted negative samples for class $i$.

For instance, consider a model that predicts crop A, which contributes 70% of the data, with an accuracy of 90%, and crop B, which contributes 30% of the data, with an accuracy of 20%.

The weighted accuracy can be calculated as follows:

$$\text{Weighted Accuracy} = (0.7 \times 0.90) + (0.3 \times 0.20) = 0.63 + 0.06 = 0.69 \text{ or } 69\%$$

**Macro-Averaged Accuracy**

Another accuracy metric is macro-averaged accuracy. It treats each class equally, regardless of its frequency. It simply averages the accuracies of each class:

$$\text{Macro-Averaged Accuracy} = \frac{0.90 + 0.20}{2} = \frac{1.10}{2} = 0.55 \text{ or } 55\%$$

The comparison between the two example calculations above showcases how weighted accuracy gives more consideration to the well-performing majority class (crop A in this case), resulting in a higher overall metric of 69%. In contrast, by treating each class equally, macro-averaged accuracy results in a lower overall metric of 55%. It does not reflect the model's strong performance on the majority class as effectively.

### 4.3.2 Macro-Averaged F1, Precision and Recall

Macro-averaged metrics on the other hand, as we've seen in the example given above, treat each class equally by averaging the metric scores for each class, regardless of the class size. This ensures that the performance on minority classes is not overshadowed by the majority classes. As such, these metrics serve as an important counterweight to weighted accuracy in judging model performance. The better the macro-averaged metrics, the less likely minority classes are to be overlooked, and the less their performance is degraded in favor of performance on majority classes.

In comparison to accuracy, the metrics F1 score, precision, and recall provide a more nuanced understanding of model performance on our imbalanced dataset. They focus on the balance between true positives and false positives/negatives.

Precision is the ratio of true positive predictions to the total predicted positives (true positives + false positives). High precision indicates that the model has a low false positive rate, which means it is precise in predicting the positive class. Precision is important for accurately identifying minority classes without inflating the number of positive predictions.

Recall (also known as sensitivity) is the ratio of true positive predictions to all actual positives (true positives + false negatives). High recall indicates that the model successfully identifies a large proportion of actual positives, which is vital for ensuring that minority class instances are not overlooked.

The F1 score, being the harmonized mean of them, balances precision and recall. This balance helps evaluate the model's performance on minority classes, where both identifying true positives and minimizing false positives are equally important.

The formulas for macro-averaged precision, recall, and F1 score are as follows:

$$\text{Macro-Averaged Precision} = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{4.7}$$

$$\text{Macro-Averaged Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} \tag{4.8}$$

$$\text{Macro-Averaged F1 score} = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4.9}$$

where:

- $N$ is the total number of classes.
- $TP_i$, $FP_i$, $FN_i$ are the true positives, false positives, and false negatives for class $i$, respectively.
- $\text{Precision}_i$ and $\text{Recall}_i$ are the precision and recall for class $i$, respectively.

### 4.3.3 Per-Class Metrics

For class-specific insights into model performance, we log a table of the metrics F1 score, recall (accuracy), and precision calculated per class. It is important to note that in a multi-class classification scenario, per-class accuracy equals per-class recall. They both represent the ratio of true positives for the given crop class against the number of actual positives for that class, with all true negatives and false positives ignored. When evaluating our model performance against the ConvSTAR benchmark by Turkoglu et al. (2021), we thus compare their per-class accuracy with our logged per-class recall, which is equal to per-class accuracy.

### 4.3.4 Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's performance, showing how many instances of each class were correctly or incorrectly classified. It is essential for identifying specific areas where the model performs well or poorly and understanding the types of errors it makes. The confusion matrix reveals which crops are misclassified as others, illuminating interesting model behaviors. For instance, it might become apparent that the class *Pasture* is often misclassified as *Meadow*. This makes sense when considering their similar visual appearance and highlights a potential flaw or ambiguity in the label hierarchy. Insights like this can guide further refinement of the model and the labeling strategy to improve overall classification accuracy. Furthermore, the confusion matrix allows to judge per-class accuracy at one glance in a visual way.

In order to facilitate easier interpretation, the rows and columns of the confusion matrix are ordered descending by class size. Additionally, the sample size $N$ for each class in the dataset, on which the matrix was calculated, is shown.

We represent the confusion matrix for our multi-class crop classification problem as follows:

$$\mathbf{C} = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,N} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N,1} & C_{N,2} & \cdots & C_{N,N} \end{bmatrix} \tag{4.10}$$

where $C_{i,j}$ is the number of instances of class $i$ (target class) that were predicted as class $j$ (predicted class). In this matri, each row represents the actual (target) classes and each column represents the predicted classes. The diagonal elements $C_{i,i}$ represent the correctly classified instances for each class, while the off-diagonal elements indicate misclassifications.

### 4.3.5 Cohen's Kappa

Cohen's kappa is a statistic for inter-rater agreement on nominal data. It serves as a measure for how much better a trained model is performing compared to a model that guesses at random according to the frequency of each class. It's values range from -1 to 1, where, 1 indicates perfect agreement, 0 indicates no agreement beyond what would be expected by chance, and values less than 0 indicate that the classifier is performing worse than random guessing (McHugh, 2012). Because it takes into consideration the expected accuracy, which is the accuracy that would be achieved by a random classifier making predictions based on the frequency of each class, it is especially useful for imbalanced datasets like ours. Cohen's kappa provides a valuable measure of the model's reliability across the different tiers of the crop classification task.

According to McHugh (2012), Cohen's kappa can be interpreted through the coefficient of determination (COD), where squaring the kappa value conceptually indicates the percentage of data reliability. For instance, a kappa value above 0.90 signifies almost perfect agreement, implying that 82-100% of the data is reliable (McHugh, 2012).

The formula for Cohen's kappa is given by:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{4.11}$$

where:

- $p_o$ is the observed agreement between the raters or classifiers.
- $p_e$ is the expected agreement by chance.

### 4.3.6 Segmentation Masks

In addition to the quantitative metrics, qualitative segmentation masks (see Figure 4.2) of the last batch of training and validation data are logged at the end of every epoch. These interactive segmentation masks consist of multiple layers that can be shown and hidden as needed. Each crop type is indicated in a specific color, that is consistent for all layers.

The layers are:

1. The input satellite image being classified (RGB-channels only)
2. The correctness of the field majority prediction, indicating correctly classified fields in green and wrongly classified fields in red
3. The ground truth fields with their target class
4. The pixel-wise class prediction
5. The field majority class prediction

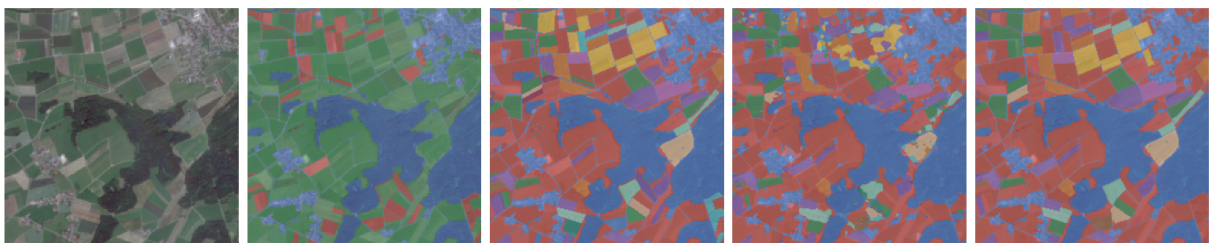


**Figure 4.2:** A segmentation mask as produced by Messis on a validation sample. Tiles shown in the order of the layer enumeration in Section 4.3.6 Segmentation Masks.

## 4.4 Selected Evaluation Metrics

With the extensive range of metrics being tracked, it's essential to narrow our focus to a select few for effectively comparing different model versions in our experiments.

We judge model performance by looking at:

- Weighted Accuracy (or Macro-Averaged Accuracy)
- Macro-Averaged F1 score
- Confusion Matrix

Note that during the first experiments, weighted accuracy was not tracked. These experiments are evaluated based on macro-averaged accuracy instead.

By using these metrics, we aim to ensure that our models not only perform well in predicting the most common crops but also maintain reasonable performance on less represented classes. In a potential applied use case, this is essential for providing a robust and practical solution for crop classification.

## 4.5 Transfer Learning

Prithvi is only the base model for feature extraction, incapable of performing classification operations independently. Consequently, it is necessary to extend the model to enable classification tasks. This entails expanding the model with several additional layers that must be learned, while maintaining the parameters of the base model. This approach, which we utilize in the architecture and training of Messis, is commonly known as transfer learning. Specifically, we added two components to the base model (the backbone described in Section 4.1.1 Backbone: Prithvi Vision Transformer):

- A neck described in Section 4.1.2 Neck: From Tokens to a Single Spatially-Structured Embedding that converts the tokens from the backbone into a single spatially-structured embedding.
- A head that converts the single spatially-structured embedding into a crop class for each pixel, as described in Section 4.1.3 Head: Spatially-Structured Embedding to Crop Classes.

Transfer learning allows a model to be trained with a much smaller set of examples. As demonstrated by Jakubik et al. (2023), just 400 chips are enough to outperform other state-of-the-art models such as a CGAN trained on more than 10 times the data on a multi-temporal cloud gap imputation task. Not only are fewer training examples needed for transfer learning. The model also needs fewer training epochs to learn, as it can build upon previously learned knowledge and features. Notably, Jakubik et al. (2023) used only half as many epochs when fine-tuning Prithvi on the flood mapping dataset Sen1Floods11[13], compared to learning from randomly initialized weights. This lowers the computational costs further.

In Experiment 5.4 Fine-tuning Prithvi, we compared this transfer learning approach to a fine-tuning approach where we unfroze the weights from the backbone and adapted them alongside the rest of the weights. In the same experiment, we also compared the transfer learning approach to a model with the same architecture that is trained from scratch with randomly initialized weights.

[13] https://github.com/cloudtostreet/Sen1Floods11

## 4.6 Cross-Validation and Data Split Strategies

To accurately estimate the uncertainty of the reported scores and ensure the evaluation is based on a test fold not used during training or hyperparameter tuning, we employed a cross-validation strategy with a train-validation-test split. The dataset was divided into 6 folds, optimizing for a uniform distribution in the number of chips across all folds. The last fold was reserved exclusively for testing purposes, while the remaining 5 folds were used for 5-fold cross-validation. This allows for a reliable estimation of the variance in the reported scores by evaluating the model's performance on multiple training-validation splits.

The split was done with 4 different strategies, as described in the following subsections. We used our stratified strategy as described in Section 4.6.3 Stratified Strategy in all experiments as default, but performed Experiment 5.1 Dataset Fold Split Strategy that compares these strategies and their efficacy.

### 4.6.1 Band Strategy

The band strategy is analogous to the strategy used by Turkoglu et al. (2021) for their ZueriCrop dataset, just with 6 instead of 5 bands. The area gets split into 6 vertical bands or strips, with the chips within each band assigned to the same fold. As the number of pixels on each side of our selected area was not divisible by $224 \times 6$ with a remainder of 0, the chips were selected in a way that centered the selected area and ensured each band had the same width. This resulted in $18 \times 22$ (396) chips and 96 pixels from the top and bottom and 544 pixels from the left and right of the selected area being discarded. Furthermore, 27 empty chips with no ground truth were discarded, leaving 369 remaining. In total, 7 699 456 pixels were discarded with this strategy. The resulting 6 bands can be seen in Figure 4.3.

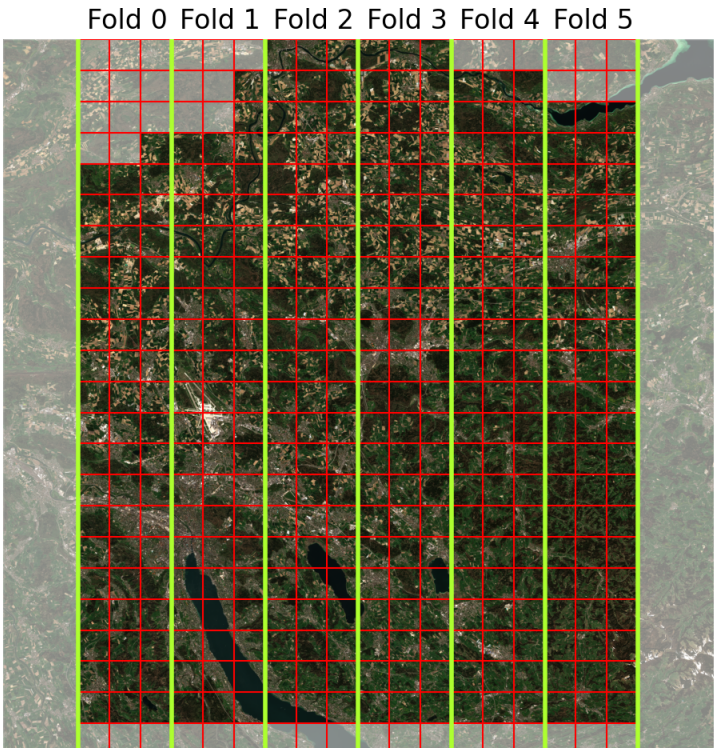



**Figure 4.3:** Selected chips with the band split strategy. Green lines mark the edges of the folds and each red square is a chip. The gray area is discarded because it contains no ground truth, does not create a full chip, or else, the bands would not be of equal width.

### 4.6.2 Sechidis Strategy

Splitting data with multiple labels and different pixel counts for each presents a significant challenge, as achieving an ideal split is not straightforward. There is no efficient method to achieve the perfect split. The only way to achieve that is by using a brute force approach and testing each combination, which naturally has a terrible runtime complexity of $O(n!)$. Sechidis et al. (2011) discuss a slightly relaxed problem when the labels don't have a pixel count, but only a flag that marks if a label is present or not.

Sechidis et al. (2011) proposed two heuristics to approximate a stratified k-fold split for multi-label data, but did not find an algorithm to find a perfect solution:

- The iterative stratification algorithm is more fit for data with approximately the same number of distinct labelsets as the number of examples (Sechidis et al., 2011).
- The labelsets-based stratification algorithm fits better to datasets with fewer distinct labelsets than examples (Sechidis et al., 2011).

Where a labelset is simply the set of labels appearing in each data point.

As our dataset has a ratio of distinct labelsets to examples of 0.886, we chose to expand their first iterative stratification algorithm to labelsets with pixel counts. The Sechidis et al. (2011) algorithm distributes examples iteratively by finding the labels with the fewest remaining examples and then assigning them to the fold with the most desire for examples of this class. The idea behind this greedy choice is that rare labels may spread in an undesirable fashion if they are not prioritized for examination, and this cannot be fixed later. However, because there are more examples available when labels are common, we may eventually be able to change the existing distribution in the direction that we want (Sechidis et al., 2011). The labelsets-based stratification algorithm was designed to work with datasets where each example can only have a label present or not. We therefore changed the algorithm such that instead of simply counting examples with a label, we now sum the total number of pixels for each label across the entire dataset. Then, we use this aggregated data to identify the label with the fewest pixels and prioritize assigning it to the fold with the lowest pixel count for that label.

Since we aren't limited by this strategy to have the number of chips in the horizontal direction divisible by 6, we were able to use the maximum number of chips that can fit into the area. Again, we couldn't use the whole area as the length of one side (5120) cannot be divided by 224 without a remainder. We therefore ensured that the selected area was centered and empty chips were discarded again. This resulted in 22x22 (484) chips of which 55 were discarded and a total of 429 chips remained. Therefore a total of 4 688 896 pixels were discarded. This resulted in an additional 3 010 560 pixels or 60 chips compared to the band strategy described in Section 4.6.1 Band Strategy. These calculations are also valid for the following two strategies.

### 4.6.3 Stratified Strategy

Our strategy aims to iteratively assign a chip to a fold by minimizing the mean kullback-leibler divergence over all folds in each step as given in the Formula 4.14, where $N$ is the number of distributions $q$ in the set $\{q_j\}$, and $M$ is the number of elements in each distribution. To avoid the metric being dominated by the few head classes with the highest pixel counts in our longtail distribution of the crop types we avoid using raw scores and therefore transform the pixel counts using the square root function and add 0.1 to the transformed score to avoid a division by 0 error as described in Formula 4.12. Furthermore, each element $p_i$ and $q_ij$ in the distributions $p$ and $q$ are the relative frequencies of the transformed pixel counts of the $i$-th class as described in Formula 4.13, where $n_i$ and $n_j$ describe the transformed pixel count in the $i$-th fold and $j$-th fold

respectively. The number of distributions is equal to the number of folds, as each fold has its own distribution. In order to ensure that the distributions in both the training and validation datasets are comparable, we want to maintain similarity between the distributions of each fold.

$$\widehat{\text{pixel\_count}} = \sqrt{\text{pixel\_count}} + 0.1 \tag{4.12}$$

$$p_i = \frac{n_i}{\sum_{j=1}^{M} n_j} \tag{4.13}$$

$$\overline{D}_{\mathrm{KL}}(P \parallel Q) = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{M} p_i \log \left( \frac{p_i}{q_{ij}} \right) \tag{4.14}$$

$$\text{penalty} = \alpha \cdot \sqrt{\sum_{i=0}^{N_{\text{folds}}} (\overline{n} - n_i)^2} \tag{4.15}$$

$$\alpha = \frac{1}{30\sqrt{n_{\text{chips}} - i + 1}} \tag{4.16}$$

Further, we add a penalty for the number of chips in each fold such that each fold tries to obtain an identical number of chips. The penalty is given in Formula 4.15, where $\overline{n}$ represents the mean over all folds of the sum of the transformed pixel counts in a fold, and $n_i$ represents the sum of the transformed pixel counts in the $i$-th fold. The penalties hyperparameter $\alpha$ is defined in the Formula 4.16. There $n_{\text{chips}}$ represents the number of all chips and $i$ denotes the current iteration number. This way the penalty increases over time and makes this approach focus on the balance of the folds at the end and not in the beginning. We experimented with different formulas for $\alpha$ as shown in Table A.5 and chose the one that minimized the mean kullback-leibler divergence.

**Algorithm 1** AssignChipsToFolds$(C, k)$

**Input** A set of unassigned chips $C$ and the number of folds $k$
**Output** A list of disjoint subsets $S_1, \ldots, S_k$ of chips

// Randomly assign one chip to each fold and remove from $C$
$S \leftarrow []$
**for** $j \leftarrow 1$ **to** $k$ **do**
    $r \leftarrow randomIndexOf(C)$
    $S_j \leftarrow S_j + C_r$
    $C \leftarrow C \setminus \{C_r\}$
**end for**
**while** $C \neq \emptyset$ **do**
    $(c^*, s^*) \leftarrow \underset{c \in C, s \in \{1,\ldots,k\}}{\text{argmin}}$ meanKLDivergence$(S_1, \ldots, S_s \cup \{c\}, \ldots, S_k)$
    $S_{s^*} \leftarrow S_{s^*} \cup \{c^*\}$
    $C \leftarrow C \setminus \{c^*\}$
**end while**
**return** $S$

The algorithm as described in Algorithm 1, starts by randomly assigning one chip to each fold. We then assign each remaining chip to each fold and then evaluate the mean kullback-leibler

divergence for each possible chip-fold combination to see how adding one chip to a fold changes the score. We then select the chip-fold combination with the lowest score and add that chip to said fold, essentially greedy assigning one chip at a time. We continue iterating until all chips are assigned. The resulting distributions of each fold can be viewed in Figure A.5.

### 4.6.4 Random Strategy

This strategy works by randomly distributing all non-empty chips among the folds, with the sole objective of ensuring that each fold ends up with an equal amount of chips.

## 4.7 Experiment Setup

This section provides a detailed overview of the experimental setup employed throughout our research. It covers the tools and methodologies used to manage and evaluate the experiments, including tracking, baseline comparisons, loss functions, optimization strategies, early stopping criteria, uncertainty estimation, and training configurations.

### 4.7.1 Weights & Biases

The Weights & Biases (W&B) Machine Learning platform is employed to track and evaluate all experiments. The detailed results of each run, along with an associated report, are publicly available[14]. The report provides an easy way to view and compare the core metrics across all experiments, with run sets for each experiment that can be activated or hidden as needed[15].

The nomenclature of the runs is as follows:
`exp-[ID]-[EXPERIMENT-GROUP]-[EXPERIMENT-NAME]-[VAL-FOLD-ID]`, where each experiment is assigned an ID, followed by the experiment group which indicates the topic of the experiment, and the specific name of that experiment which hints at what it was about within the experiment group. The Validation fold ID specifies which fold was used for validation in that run, with all other folds, except fold 6 (which is exclusively reserved for testing), being used as training folds. This is relevant in the context of the 5-fold Cross-Validation that is often performed. To examine particular experimental outcomes in full detail, it is possible to filter the W&B runs using the experiment key provided in the caption of each result table that is listed in Section 5 Experiments and Results.

### 4.7.2 Loss Function

The loss function we use for Messis is a weighted sum of cross-entropy losses. We assign a weight $\lambda_t$ to each hierarchical head that determines how much influence each tier $t$ of the label hierarchy has during training. The refined predictions from the refinement module are weighted by an additional term $\gamma$. These weights ensure that the network considers the hierarchical relationships between labels without enforcing them strictly, and guide the model to learn features that respect the hierarchy. For example, features that help identify specific types of orchards, like apple and pear orchards, also help in classifying them under a general orchard category. The number of tiers in the label hierarchy is given by $T$.

$$L = \sum_{t=1}^{T} \lambda_t \mathrm{CE}\left(Y^t, \hat{Y}^t\right) + \gamma \mathrm{CE}\left(Y^T, \hat{Y}^T_{\text{refined}}\right) \tag{4.17}$$

[14] wandb.ai/crop-classification/messis
[15] https://wandb.ai/crop-classification/messis/reports/Experiment-Overview–Vmlldzo4MzAzNTI1

### 4.7.3 Optimizer

We utilize Adam Optimizer with a learning rate of $1e-3$ and no weight decay, if not stated otherwise. Experiment 5.2 Comparing Optimizers focuses on comparing model performance with different optimizers and settings.

### 4.7.4 Early Stopping

Qualitative evaluation of initial training runs indicated that the target metrics, F1 score, and accuracy, continued to improve even after the validation loss reached its minimum. Consequently, we made the decision to train all models with early stopping, based on the maximum value of the metric *val_f1_tier3_majority*, with a patience of 10.

### 4.7.5 Estimation of Uncertainty in Evaluation

As mentioned in the section before, early stopping with a patience of 10 epochs was employed in all experiments. Consequently, the reported metrics are taken from the epoch where the target metric *val_f1_tier3_majority* achieves its maximum value for each run. The reported metrics are averaged across all cross-validation folds, and the standard deviation reflects the variability of the scores at these maximum epochs across different folds.

As we did not have unlimited resources, we could not do a 5-fold cross-validation for all experiments (specifically Experiment D.3 More Channels in Hierarchical Heads and Experiment D.4 More Convolutional Layers in Hierarchical Heads). To still be able to compare the setups in the experiments ran on just 1-fold using the variability of the metrics, we used the average standard deviations from other experiments. We obtained these standard deviations as follows:

1. We first selected the experiments where all setups had the same recorded metrics as the experiments we could only run 1-fold. Specifically, the weighted accuracy (W. Acc.) and F1 score over the tiers $T_1$, $T_2$, and $T_3$.
2. We discarded Experiment 5.12 Best Model: Messis, as the 1-fold experiments influenced its hyperparameters. This left us with the experiments:
   - 5.5 Improving Generalization with Data Augmentation
   - 5.6 Ignoring the Background in Loss Calculation
   - 5.8 Dropping the Hierarchy
   - 5.9 Kernel Size in Hierarchical Head
   - 5.10 Parameter Efficiency in the Neck
   - 5.11 More Timesteps
   - D.2 Impact of Dropout Type and Probability
3. We then selected all setups with 5 folds from these experiments. This left us with 15 setups, for which we computed the mean of each metric's standard deviation using the Formula 4.18 for equal sample sizes.

The resulting mean standard deviations can be seen in Table 4.1.

$$\text{Mean S.D.} = \sqrt{\frac{s_1^2 + s_2^2 + \cdots + s_k^2}{k}} \tag{4.18}$$

| Metric | mean standard deviation |
|---|---|
| F1 $T_1$ | 5.4 |
| F1 $T_2$ | 1.6 |
| F1 $T_3$ | 1.0 |
| W. Acc. $T_1$ | 0.3 |
| W. Acc. $T_2$ | 0.9 |
| W. Acc. $T_3$ | 1.1 |

**Table 4.1:** The mean standard deviation per tier, computed over 15 experiment setups.

### 4.7.6 Lightning Trainer Setup

We selected a batch size of 16 for all experiments and maintained this consistently throughout. In cases where memory constraints prevented maintaining the batch size of 16, we employed gradient accumulation to effectively simulate this batch size. All models were trained on a single GPU using mixed precision (16-bit). The training process was managed using the PyTorch Lightning Trainer framework.

### 4.7.7 Baselines

For each experiment group, we establish a baseline by training the model with the standard setup, excluding the specific changes introduced by the experiment. Each experiment is then compared to its baseline to evaluate the impact of the experimental changes. Essentially, the baseline represents the model performance with the standard hyperparameters, data, and label hierarchy, against which alternative approaches within the experiment group are assessed. The standard model setup is Messis as it was introduced in Section 4 Methodology. It includes the use of the ZueriCrop 2.0 dataset with the original ZueriCrop Label Hierarchy, created based on our Kullback-Leibler divergence minimization data splitting strategy. All the changes performed to that standard setup are explained for each respective experiment in subsections in Section 5 Experiments and Results.

# 5 Experiments and Results

In this section, we describe each experiment we conducted to answer our RQs and explore Prithvi's limits, possibilities, and obstacles during the fine-tuning on the ZueriCrop 2.0 dataset. We did not deem all experiments relevant enough to be included in this section. These experiments can be found in Appendix D Additional Experiments and Results along with their results. Each subsection introduces the experiment, defines the objective, lists the setups, presents the results, reports key findings, and draws a short conclusion.

## 5.1 Dataset Fold Split Strategy

In this experiment, we aim to evaluate different strategies for splitting the dataset into folds, which is crucial for assessing the variability of reported metrics through cross-validation. As outlined in Section 4.6 Cross-Validation and Data Split Strategies, we split the dataset into six different folds using 4 different strategies. We created these approaches as there is no state-of-the-art strategy on how to split datasets with pixel counts per class per data point in different folds. We compare these 4 strategies in this experiment to determine the most effective way to ensure reliable performance across different folds and achieve a more balanced and representative data distribution across them. The results are reported in Table 5.1.

Experiment Setups:

- **random (5-fold)**: Each chip is randomly assigned to a fold, ensuring that each fold contains an equal number of chips, as described in Section 4.6.4 Random Strategy. This method does not account for class distribution, potentially leading to imbalanced folds (baseline).
- **band (5-fold)**: The area is split into six vertical bands, with each band assigned to a different fold as described in Section 4.6.1 Band Strategy. This method was used by Turkoglu et al. (2021) in their ZueriCrop dataset and aims to maintain spatial continuity but may result in uneven class distributions.
- **sechidis (5-fold)**: Sechidis et al. (2011) proposed to distribute chips iteratively. Each step starts by first identifying the class with the fewest pixels. The chip with the most pixels of this class is then assigned to the fold with the least pixels for this class, as described in Section 4.6.2 Sechidis Strategy. This method aims to balance class distributions across folds.
- **stratified (5-fold)**: Our strategy also works iteratively, but by minimizing the mean Kullback-Leibler divergence of all folds at each step, as described in Section 4.6.3 Stratified Strategy. This method also aims to balance class distributions across folds.

| Metric | † random (5-fold) | band (5-fold) | sechidis (5-fold) | stratified (5-fold) |
|---|---|---|---|---|
| F1 $T_1$ | 44.2% ± 5.3% | 43.0% ± 5.2% | 44.1% ± 6.5% | **47.6% ± 7.7%** |
| F1 $T_2$ | 28.8% ± 0.9% | **30.9% ± 1.6%** | 29.4% ± 1.4% | 30.7% ± 2.1% |
| F1 $T_3$ | 17.2% ± 1.8% | 18.2% ± 2.1% | **18.2% ± 1.3%** | 18.0% ± 1.9% |
| Acc. $T_1$ | 44.2% ± 5.9% | 43.5% ± 6.0% | 45.0% ± 5.9% | **46.6% ± 6.3%** |
| Acc. $T_2$ | 28.3% ± 1.4% | **30.7% ± 2.0%** | 29.4% ± 1.1% | 29.8% ± 1.9% |
| Acc. $T_3$ | 16.8% ± 1.7% | 17.8% ± 2.0% | **17.9% ± 0.9%** | 17.3% ± 1.7% |

**Table 5.1:** Results for experiment exp-1-dataset-split-strategy. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Findings:

1. **Baseline Performance (random)**: The random split strategy yielded the worst performance overall. This approach likely resulted in the least balanced folds leading to suboptimal model evaluation performance.
2. **Band Split**: This strategy performed best for the F1 $T_2$ score (30.9%) and accuracy $T_2$ (30.7%), indicating spatial continuity might benefit mid-tier classifications. However, the scores were not better by one standard deviation than the second-best results.
3. **Sechidis Split**: The sechidis strategy achieved the best accuracy on Tier 3 (17.9%), but only by +0.1pp compared to the second-best. This is insignificant when compared to the standard deviations.
4. **Stratified Split**: The stratified split strategy showed the best performance on Tier 1 but failed to do so significantly when comparing the difference to the second-best scores to their standard deviations.

The findings indicate that the random split strategy is suboptimal for our task, as it had the overall lowest performance. Each of the more sophisticated strategies (band, sechidis, and stratified) demonstrated strengths in specific tiers, though none dominated across all metrics. Each domination is insignificant when relating their advantage over the second-best scores to their standard deviations. Thus, we continue utilizing our stratified approach in further experiments.

## 5.2 Comparing Optimizers

The goal of this experiment is to evaluate the impact of different optimizers and learning rates on the performance of the model. Optimizers play a crucial role in the training process by adjusting the weights to minimize the loss function. We determine which optimizer configuration yields the best performance across our metrics. The results are shown in Table 5.2

Adam is widely considered one of the most established and commonly used optimizers today. It is an optimization algorithm that combines aspects of Momentum and RMSProp, adapting the learning rate for each parameter individually. The combination of Momentum and RMSProp allows Adam to be highly efficient and robust, making it often the first choice for deep learning models (Kingma & Ba, 2017). Therefore, we consider Adam as our baseline for this experiment.

AdamW is a variant of Adam that decouples weight decay from the gradient update. Thus, for AdamW, weight decay is equivalent to L2 regularization, which is motivated by the observation that networks with smaller weights are less prone to overfitting and generalize better. For this reason, AdamW typically leads to better generalization performance and improved training stability (Loshchilov & Hutter, 2019).

Mini-Batch Gradient Descent (MBGD) is a fundamental algorithm based on the Gradient Descent (GD) algorithm that is more computationally efficient, using batches to approximate gradient updates instead of the full dataset. It uses the same learning rate on all parameters. While basic Mini-Batch Gradient Descent (MBGD) is less susceptible to getting stuck in local minima of the loss landscape compared to Full-Batch Gradient Descent due to the stochasticity introduced by mini-batches, on its own, it can still be less effective for deep learning. Enhancements like momentum, which we utilize, can help accelerate convergence and navigate through loss valleys more effectively (Ruder, 2017).

While basic Mini-Batch Gradient Descent (MBGD) is somewhat better at escaping local minima in the loss landscape compared to Full-Batch Gradient Descent due to the stochasticity introduced by mini-batches, techniques like momentum can further enhance this capability.

As a fourth and last optimizer, we experiment with the recently proposed Lion. It has been shown to improve convergence speed and performance on specific tasks, such as a ViT on ImageNet, by up to 2%. It is more memory-efficient than Adam as it only keeps track of the momentum, and shows similar or better performance (Chen et al., 2023). This gain in memory-efficiency is particularly relevant in large-scale models. In practice, reducing the learning rate compared to Adam by a factor of 3x/10x and increasing weight decay by 3x/10x respectively is recommended by (Wang, 2024). One potential limitation of Lion is that its performance gains were shown to grow with the training batch size (Chen et al., 2023).

Experiment Setups:

- **adam-1e-3 (5-fold)**: Adam optimizer with a learning rate of $1 \times 10^{-3}$, no weight decay (baseline).
- **adamw-1e-3 (5-fold)**: AdamW optimizer with a learning rate of $1 \times 10^{-3}$ and weight decay $1 \times 10^{-2}$.
- **lion-10 (5-fold)**: Lion optimizer with a learning rate of $1 \times 10^{-4}$ and weight decay of $1 \times 10^{-1}$.
- **sgd-momentum (5-fold)**: MBGD with momentum optimizer, using learning rate $1 \times 10^{-3}$ and momentum 0.9.

| Metric | † adam-1e-3 (5-fold) | adamw-1e-3 (5-fold) | lion-10 (5-fold) | sgd-momentum (5-fold) |
|---|---|---|---|---|
| F1 $T_1$ | **48.2% ± 4.7%** | 46.9% ± 4.9% | 42.7% ± 4.2% | 42.5% ± 6.2% |
| F1 $T_2$ | **28.6% ± 0.8%** | 27.3% ± 0.8% | 23.7% ± 2.5% | 24.6% ± 2.7% |
| F1 $T_3$ | **14.6% ± 0.3%** | 13.9% ± 0.5% | 10.4% ± 1.8% | 12.2% ± 1.3% |
| Acc. $T_1$ | 46.4% ± 4.7% | 47.0% ± 4.4% | **48.6% ± 6.2%** | 41.8% ± 5.9% |
| Acc. $T_2$ | **27.8% ± 0.7%** | 27.3% ± 0.7% | 26.1% ± 1.9% | 24.7% ± 2.4% |
| Acc. $T_3$ | **15.0% ± 0.5%** | 14.2% ± 0.5% | 12.0% ± 1.3% | 12.8% ± 1.2% |

**Table 5.2:** Results for experiment exp-2-optimizer. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Findings:

1. Baseline Performance (adam-1e-3): The baseline setup using Adam with a learning rate of $1 \times 10^{-3}$ achieved the highest F1 scores for all tiers and accuracy for Tier 2 and 3, demonstrating strong overall performance.
2. AdamW Performance: The AdamW optimizer with the same learning rate and additional weight decay performed second best after the baseline in F1 scores and accuracy. This indicates it is a viable alternative with a slightly different regularization effect.
3. Lion Optimizer: The Lion optimizer with a learning rate of $1 \times 10^{-4}$ and weight decay of $1 \times 10^{-1}$ showed lower performance across most metrics, except for Tier 3 accuracy. An additional test using 3x decreased learning rate and 3x increased weight decay compared to the Adam setup terminated early due to the validation loss becoming NaN (Not a Number), indicating numerical instability. This would require further investigation, but is out of scope for our work.
4. MBGD with Momentum: The MBGD with momentum configuration showed lower performance in both accuracy and F1 scores, compared to Adam and AdamW. This highlights the importance of adaptive learning rates provided by Adam-type optimizers.

Adam and AdamW demonstrated the fastest loss convergence, followed by Lion and MBGD. Overall, the Adam optimizer with a learning rate of $1 \times 10^{-3}$ remains the best-performing configuration for Messis. It balances high performance with stability across the evaluated metrics.

## 5.3 Satellite Imagery Time Ranges

With this experiment, we evaluate different satellite imagery time ranges. Prithvi's input is limited to 3 timesteps/images. There are options to get around the 3-timesteps limit of Prithvi, such as passing groups of 3 timesteps through the model individually, as explored in Experiment 5.11 More Timesteps.

This limits us to 3 satellite images from the 142 theoretically available within the time range covered by ZueriCrop. As a baseline for this and all other experiments, if not specifically mentioned otherwise, we used the same approach as Jakubik et al. (2023). They selected the first, last, and middle image in the time range from March to September where cloud cover was below 5%.

However, not all crops are planted in March, and some are already harvested before September (Patricia Rutz, 2023). Accordingly, the satellite images from March, September, and the one in between might not observe any relevant crop activity and therefore lack important information for the model. It is crucial to consider which timesteps provide the most information to the model. Gregor Perich (2024a) recommended additionally exploring the time ranges from April to August and May to August. Therefore, we compared the three time ranges March to September, April to August, and May to August, while continuing to select the first, middle, and last image with a cloud cover of 5% or lower in each range. The results can be seen in Table 5.3.

Experiment Setups:

- **march-september (5-fold)**: Three images between 1st of March and 30th September 2019 with less than 5% cloud cover: 21.03.2019, 26.06.2019, and 29.09.2019 with a time interval of 97 and 95 days (baseline).
- **april-august (5-fold)**: Three images between 1st of April and 31st August 2019 with less than 5% cloud cover: 20.04.2019, 26.06.2019, and 30.08.2019 with a time interval of 67 and 65 days.
- **may-august (5-fold)**: Three images between 1st of May and 31st August 2019 with less than 5% cloud cover: 26.06.2019, 24.07.2019, and 30.08.2019 with a time interval of 28 and 37 days.

| Metric | † march-september (5-fold) | april-august (5-fold) | may-august (5-fold) |
|---|---|---|---|
| F1 $T_1$ | 47.2% ± 7.1% | **49.7% ± 3.9%** | <u>47.4% ± 4.3%</u> |
| F1 $T_2$ | 29.1% ± 1.8% | **31.1% ± 0.8%** | <u>30.9% ± 1.3%</u> |
| F1 $T_3$ | 19.4% ± 0.2% | **21.7% ± 1.3%** | <u>20.1% ± 0.6%</u> |
| Acc. $T_1$ | 46.2% ± 8.6% | **50.6% ± 5.0%** | <u>47.7% ± 3.9%</u> |
| Acc. $T_2$ | 28.2% ± 2.3% | **30.8% ± 1.3%** | <u>30.4% ± 1.3%</u> |
| Acc. $T_3$ | 18.7% ± 0.3% | **20.9% ± 1.0%** | <u>19.4% ± 1.0%</u> |

**Table 5.3:** Results for experiment exp-3-timeranges. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Findings:

1. **Baseline Performance (March-September)**: The baseline configuration utilizing images from March to September performed the worst across all reported metrics.

2. **Shorter time range (April-August)**: The setup using images from April to August outperformed the baseline significantly. It achieved the highest scores for F1 $T_1$ (49.7%), F1 $T_2$ (31.1%), and F1 $T_3$ (21.7%). Additionally, it had the highest accuracy scores for all tiers: Acc. $T_1$ (50.6%), Acc. $T_2$ (30.8%), and Acc. $T_3$ (20.9%).
3. **Shortest time range (May-August)**: The setup using images from May to August achieved second-best performance metrics, surpassing the baseline but falling short of the April-August configuration in all reported metrics.

The findings highlight that limiting the satellite imagery time range to April to August yields the most informative data for our crop classification task, leading to improved classification performance compared to the broader March to September, as well as the narrower May to August range.

## 5.4 Fine-tuning Prithvi

In all experiments conducted so far, we have employed a transfer learning approach as described in Section 4.5 Transfer Learning, utilizing the pre-trained Prithvi model as the backbone for our crop classification task. The backbone weights were frozen, and we trained only the neck and head.

In this experiment, we take a further step by fine-tuning the Prithvi backbone weights. This involves unfreezing the backbone layer and continuing to train it on our dataset, allowing the model to adapt to the crop classification task. While fine-tuning can lead to improved performance by allowing the entire network to learn from the new data, it requires more computational resources and training time, as the gradients need to be computed for all parameters in the backbone layers too.

Besides transfer learning and fine-tuning, a further option is to train the model from scratch. This can serve as a reference point for whether the pre-trained weights in the Prithvi backbone are helpful for model performance. Instead of using the pretrained Prithvi weights, we randomly initialize them in all layers of the backbone using Xavier uniform initialization. Xavier Initialization creates a distribution of activations that is similar across layers, helping to maintain the variance of activations and prevent gradients from exploding or vanishing throughout the network (Glorot and Bengio, 2010). The results of this experiment can be seen in Table 5.4.

Experiment Setups:

- **frozen (5-fold)**: Transfer learning, all Prithvi backbone weights frozen (baseline).
- **not_pretrained (5-fold)**: Model trained from scratch without any transfer learning, randomly initialized weights in all layers.
- **unfrozen (5-fold)**: Fine-tuning, all Prithvi backbone weights unfrozen.

| Metric | † frozen (5-fold) | not_pretrained (5-fold) | unfrozen (5-fold) |
|---|---|---|---|
| F1 $T_1$ | **48.6% ± 6.0%** | 34.5% ± 7.1% | 47.0% ± 6.9% |
| F1 $T_2$ | **28.9% ± 1.4%** | 19.4% ± 5.4% | 28.5% ± 1.5% |
| F1 $T_3$ | 16.8% ± 1.5% | 8.8% ± 3.2% | **18.4% ± 0.6%** |
| Acc. $T_1$ | **50.1% ± 8.9%** | 34.4% ± 6.9% | 47.2% ± 8.5% |
| Acc. $T_2$ | **29.5% ± 2.3%** | 19.5% ± 5.0% | 28.1% ± 2.3% |
| Acc. $T_3$ | 17.0% ± 0.4% | 9.2% ± 2.9% | **18.0% ± 1.1%** |

**Table 5.4:** Results for experiment exp-4-backbone. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Insights:

1. **Baseline Performance (frozen)**: The baseline setup with the frozen Prithvi backbone got the highest F1 and accuracy scores for Tier 1 and Tier 2, indicating strong performance even when only the classification layers were trained, essentially applying transfer learning.
2. **Training from Scratch (not_pretrained)**: The model trained from scratch without any transfer learning performed the worst across all metrics and tiers. This setup also exhibited the highest standard deviations, which indicates significant instability and inconsistency in performance.
3. **Fine-Tuning (unfrozen)**: The fine-tuned setup (unfrozen backbone) achieved the highest scores for Tier 3 in both F1 (18.4%) and accuracy (18.0%), a +9.6pp and +8.8pp increase respectively compared to training from scratch. This almost doubling of performance demonstrates Prithvi's effectiveness in learning complex patterns for less common classes too. While it did not surpass the baseline in Tier 1 and Tier 2 for both accuracy and F1, it performed comparably with second-best scores, indicating that it can still maintain high performance while potentially providing better generalization to minority classes. It is important to note that this setup significantly increases VRAM consumption, as the gradients for weight updates must be calculated on the backbone as well.

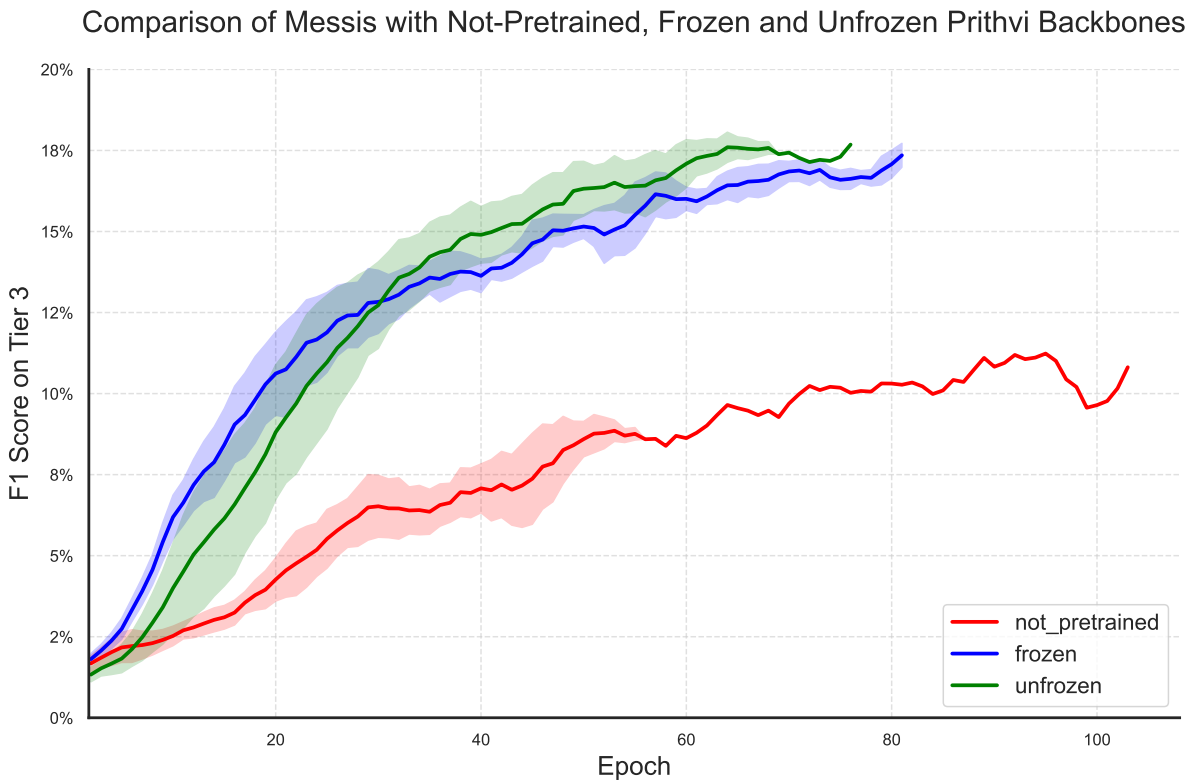



**Figure 5.1:** Comparison of the three Messis model setups using the Prithvi backbone in not-pretrained, frozen (transfer learning), and unfrozen (fine-tuning) configurations. All metrics calculated on field majority during validation phase. Rolling mean (window size = 5 epochs) reported for each setup. Shaded areas represent ±1 standard deviation. When runs end, only remaining runs contribute, causing the uncertainty band to narrow or disappear. Final 10 epochs trained due to early stopping patience are excluded.

The results, visually presented in Figure 5.1, show that both the transfer learning and fine-tuning approaches significantly increase model performance, compared to training from scratch.

These approaches leverage the extensive pretraining on general geospatial data, which enables effective training even with the limited data available in the ZueriCrop 2.0 dataset. Prithvi's ability to encode essential features of geospatial data is thus demonstrably highly relevant and helpful for our crop classification task. Remarkably, this performance improvement is achieved despite the pretraining data being from a different country and continent (the US instead of Switzerland) and having a lower pixel resolution (30x30 meters).

Overall, the unfrozen setup demonstrated the best performance in Tier 3 for both accuracy and F1 score, while remaining highly competitive for the higher tiers.

## 5.5 Improving Generalization with Data Augmentation

In this experiment, we augment our ZueriCrop 2.0 dataset during training. Data augmentation is effective against a multitude of issues in training deep learning models. For one, data augmentation can mitigate overfitting by training on a more varied dataset, which is especially helpful in scenarios where data is scarce. Similarly, generalization and robustness can be improved as the model becomes invariant to transformations like flipping and random noise (Oubara et al., 2022). The results are provided in Table 5.5.

The number of transformations we can use to augment our satellite data is limited, as for example altering the image dimension, shifting, or rotatating randomly is either impossible due to model constraints, very time-consuming to implement, or would cause a loss of usable chips. Each pixel in our dataset has specific geospatial significance, and any transformation that distorts this relationship would render the data unusable for our purposes. Therefore, we must carefully select augmentations that preserve the integrity and spatial accuracy of the satellite images. Flipping the satellite imagery horizontally and vertically with a certain probability fulfills this condition, as long as the same transformation is applied to the target mask. Another augmentation that can be utilized easily is jitter - introducing random noise to the pixel values, drawn from a normal distribution with a mean of zero and a specified standard deviation.

Experiment Setups:

- **no-aug (5-fold)**: Baseline setup with no data augmentation.
- **f-0.5-j-0.01 (5-fold)**: Data augmentation with a horizontal and vertical flip probability of 0.5 each, as well as a jitter with a standard deviation of 0.01 added to the satellite imagery bands.

| Metric | † no-aug (5-fold) | f-0.5-j-0.01 (5-fold) |
|---|---|---|
| F1 $T_1$ | 46.4% ± 6.7% | **53.1% ± 7.9%** |
| F1 $T_2$ | 31.0% ± 1.8% | **34.1% ± 2.1%** |
| F1 $T_3$ | 20.8% ± 0.7% | **23.8% ± 1.4%** |
| W. Acc. $T_1$ | **93.1% ± 0.1%** | 93.1% ± 0.3% |
| W. Acc. $T_2$ | 81.3% ± 1.1% | **82.3% ± 0.6%** |
| W. Acc. $T_3$ | 77.0% ± 1.1% | **78.6% ± 0.7%** |

**Table 5.5:** Results for experiment exp-5-data-augmentation. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Insights:

1. **Baseline Performance (no-aug)**: The baseline setup without any data augmentation achieved lower F1 and accuracy scores across all tiers compared to the augmented setup.

2. **Data Augmentation (f-0.5-j-0.01)**: The augmented setup outperformed the baseline in almost all metrics. F1 scores improved notably for Tier 1 (+6.7pp), Tier 2 (+3.1pp), and Tier 3 (+3pp). Weighted accuracy also improved with data augmentation in Tier 2 (+1pp) and Tier 3 (+1.6pp). This shows that the model benefits significantly from augmentation techniques, as they allow the model to generalize better on unseen data.
3. **Variability and Stability**: The standard deviations were slightly higher for the augmented setup in some metrics. However, this can be explained by the active introduction of more randomness to the training data.

The results show that data augmentation techniques, specifically flipping and jitter, do in fact increase model performance compared to the baseline without augmentation. Overall, the augmented setup (f-0.5-j-0.01) demonstrated better performance than the baseline in all tiers for both accuracy and F1 scores and showed that data augmentation is an effective tool for improving Messis.

## 5.6 Ignoring the Background in Loss Calculation

In this experiment, we explore the impact of setting the loss weight for the background class to zero. Currently, by the nature of the task we give it, the model inherently has to learn an arbitrary distinction between what is background and what are agriculturally used areas for which we have labels. This is a problem because the field polygons provided by the cantons, which define where what kind of crops grow, do not have labels for every area. For example, there is a significant amount of forest areas not labeled as such.

The objective is to avoid penalizing the model for incorrect background predictions, which are of no real value for classifying crops, as we have polygons describing the field boundaries. We aim to assess if this adjustment improves overall performance. Our intuition was that by removing these contradictory labels on classes such as forest, the model performance in predicting them could increase, and they might even be identified in unlabeled areas of the satellite imagery.

Experiment Setups:

- **consider (5-fold)**: Standard training with the background class included in the loss calculation (baseline).
- **ignore (5-fold)**: Training with the background class excluded from the loss calculation.

| Metric | † consider (5-fold) | ignore (5-fold) |
|---|---|---|
| F1 $T_1$ | 46.0% ± 3.6% | **49.0% ± 6.3%** |
| F1 $T_2$ | 31.0% ± 1.2% | **32.5% ± 1.7%** |
| F1 $T_3$ | 20.4% ± 1.2% | **21.0% ± 1.4%** |
| W. Acc. $T_1$ | 93.1% ± 0.2% | **93.5% ± 0.3%** |
| W. Acc. $T_2$ | **82.0% ± 0.3%** | 82.0% ± 0.8% |
| W. Acc. $T_3$ | 77.2% ± 0.7% | **77.8% ± 1.0%** |

**Table 5.6:** Results for experiment exp-6-background-loss. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Insights:

1. **Baseline Performance (consider)**: The baseline setup with the background class included in the loss calculation achieved decent performance across all tiers. Qualitative

analysis showed that with background loss taken into account, Messis learned to predict forests, roads, streets, and other infrastructure largely as background.

2. **Excluding Background Class (ignore)**: The setup with the background class excluded from the loss calculation showed improved performance in most metrics. F1 scores increased +3.0pp for Tier 1, +1.5pp for Tier 2, and +0.6pp for Tier 3. Weighted accuracies also saw slight improvements, except for Tier 2. This suggests that in our use case, not penalizing the model for incorrect background predictions leads to better classification of actual crop classes.
3. **Background Prediction Behavior**: When the background is ignored in the loss calculation, the model predicts almost no background, as it gets no positive reinforcement for classifying a pixel as such. Instead, the background areas are classified as other classes. The majority of pixels is assigned to grassland, which replaces background as the majority class, with smaller portions assigned to what appear to be other majority classes. This indicates that the model reallocates what it previously considered background into common crop categories.
4. **Noisier Predictions**: The segmentation masks reveal that what appear to be forests on the satellite image now consists of a colorful mix of classes, and the predictions appear less cohesive compared to the baseline. Previously, field-like structures of the same class and their edges were clearly defined with sharp separations and minimal overlap. Now, these edges tend to be more blurred and less distinct. One example of this is shown in Figure 5.3. However, the model begins to accurately predict forests that were previously classified as background due to inconsistent labeling in the dataset, with an accuracy of 0.16%, as can be seen in the confusion matrices shown in Figure 5.2. This highlights that the expected benefit of excluding background, allowing the model to better identify actual crop types in previously ambiguous regions, materialized.
5. **Majority Class Dominance**: A significant side effect is that the model now frequently predicts majority classes in areas previously classified as background. This includes structures like houses and roads, making the predictions in these areas of little use. The pixel-wise confusion matrix in Figure 5.2 shows this clearly when we analyze the background row. While this might seem problematic, in practical use, the model's predictions are confined within predefined polygons, so the majority-class predictions are not affected. This is evidenced by the observed increase in performance.
6. **Cluster Artifacts**: Qualitative analysis of the segmentation masks (see Figure 5.3) reveals intriguing, seemingly systematic clusters of certain classes appearing in background areas. The arrangement and spacing of these clusters suggest a link to the 16x16 pixel image patches used by the Vision Transformer. While determining the exact cause of this phenomenon would be valuable, it would require further investigation and this falls outside the scope of our work.

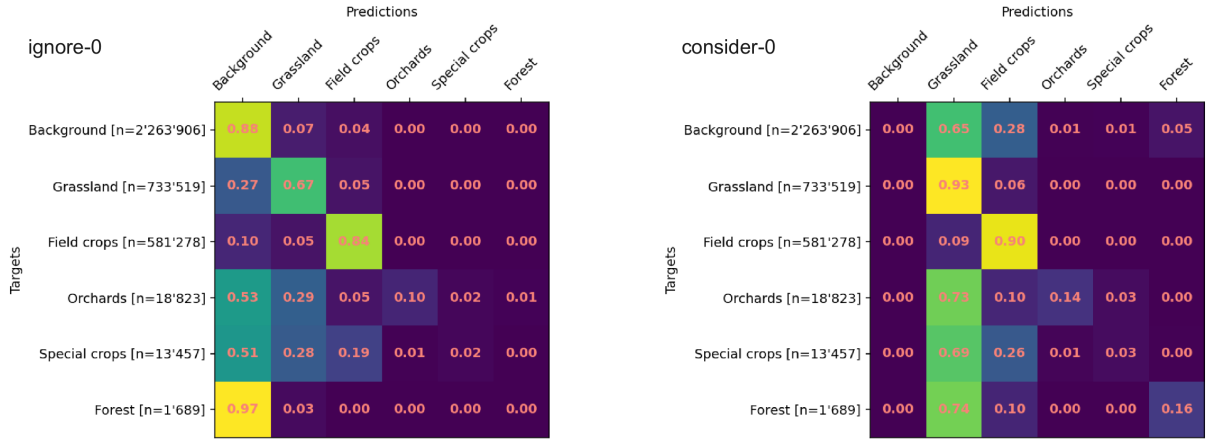



**Figure 5.2:** Validation confusion matrices in Tier 1 for both setups, calculated pixel-wise on fold 1.

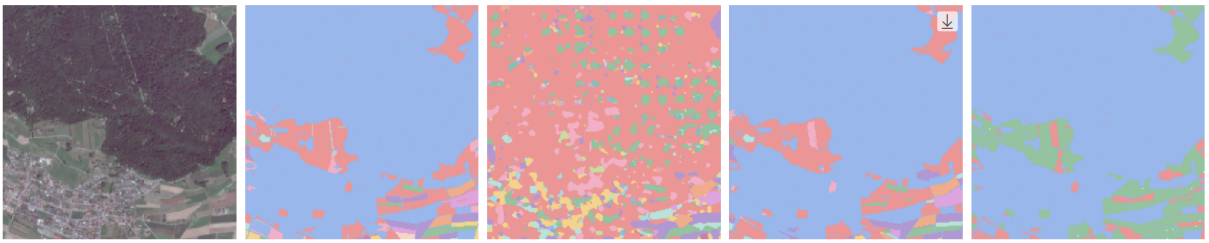


**Figure 5.3:** Cluster Artifacts appearing in pixel-wise predictions on validation data (Fold 1). From left to right: Satellite image, ground truth, pixel-wise prediction with artifacts, field majority prediction, correctness.

The results demonstrate that excluding the background class from the loss calculation improves the model's ability to classify crops correctly. This adjustment reduces the arbitrary distinctions the model has to make regarding background areas and focuses its learning on actual crop types and land area where we have meaningful ground truth. Despite some noisier pixel-wise predictions and the new dominance of the meadow class in predictions, the overall performance gains show that when using majority voting, this approach is beneficial for improving crop classification accuracy.

## 5.7 Alternative Label Hierarchy

In this experiment, we explore the impact of using an alternative label hierarchy based on the seasonality of crops. From an agricultural viewpoint, crop types can be broadly categorized into three categories based on the seasonality: 1.) annual crops planted in spring and harvested in late summer of the same year, 2.) perennial crops planted in the winter of one year and harvested in autumn of the following year and 3.) permacultures, which are grown year-round in the same location. This hierarchy was chosen to provide a more agricultural viewpoint on the classes compared to ZueriCrop (and ZueriCrop 2.0), which was focused on an arbitrary classification based on the visual similarity of crop classes as seen by an aerial sensor (e.g. the class BroadLeafRowCrops comprises crops with broad leaves grown in rows such as Rapeseed and Tobacco). The new seasonality hierarchy reduces the number of tiers and classes.

1. **Tier 1**: Annual Crop, Perennial Crop, Permaculture.
2. **Tier 2**: A total of 42 crop classes based on the seasonality, see Figure A.2.

In the transition to the seasonality hierarchy, we gain 32,290 crop pixels (about 0.4%) because where the original ZueriCrop 2.0 hierarchy only considers 80 LNF Codes, the new seasonality-based hierarchy considers 133 (+53 LNF Codes). For an introduction to LNF Codes, refer to Section 3.2 Agriculture Data. Additionally, the number of classes decreased from 48 to 42, mainly by grouping LNF Codes that represent slight variations of the same crop type. It is worth noting that the Forest class, which comprised 0.16% of the data in the ZueriCrop hierarchy, is no longer included in this revised hierarchy. Appendix A.2 ZueriCrop 2.0 Dataset with Seasonality Label Hierarchy provides a sankey diagram, detailed class distribution, and a hierarchy table, including LNF Codes, for the seasonality hierarchy.

As previously stated, the significant class imbalance represents a substantial vulnerability of the model. For example, the minority crop Chestnut has only 11 pixels in the whole dataset. This is insufficient for training the model to learn this specific class. Additionally, both when using the seasonality hierarchy and the ZueriCrop hierarchy, some minority classes are represented in only one fold, as Figure A.5 in the appendix shows. Consequently, if this fold is used as a validation or test fold, the model will not have received any training on these classes. We therefore introduce an adaption of the seasonality-based hierarchy with a minimum threshold of 10'000 pixels, below which a class will not be considered. The 10'000 pixel limit is not arbitrary; quantitative analysis of per-class metrics of our previous experiments has demonstrated that for sample sizes above 10'000 pixels, most per-class F1 scores are greater than 0. This novel hierarchy is represented in this experiment as the reduced seasonality hierarchy, and shown in Figure 5.4, with a visualization of the class distribution and the hierarchy table available in Apppendix A.3. The hierarchy consists of 24 classes, which collectively cover 104 LNF codes.

Experiment Setups:

- **zueri_crop (5-fold)**: Baseline setup using the original ZueriCrop 2.0 hierarchy.
- **seasonality (5-fold)**: New setup based on the seasonality-based hierarchy (42 classes).
- **seasonality-reduced (5-fold)**: New setup based on the reduced seasonality-based hierarchy (24 classes).

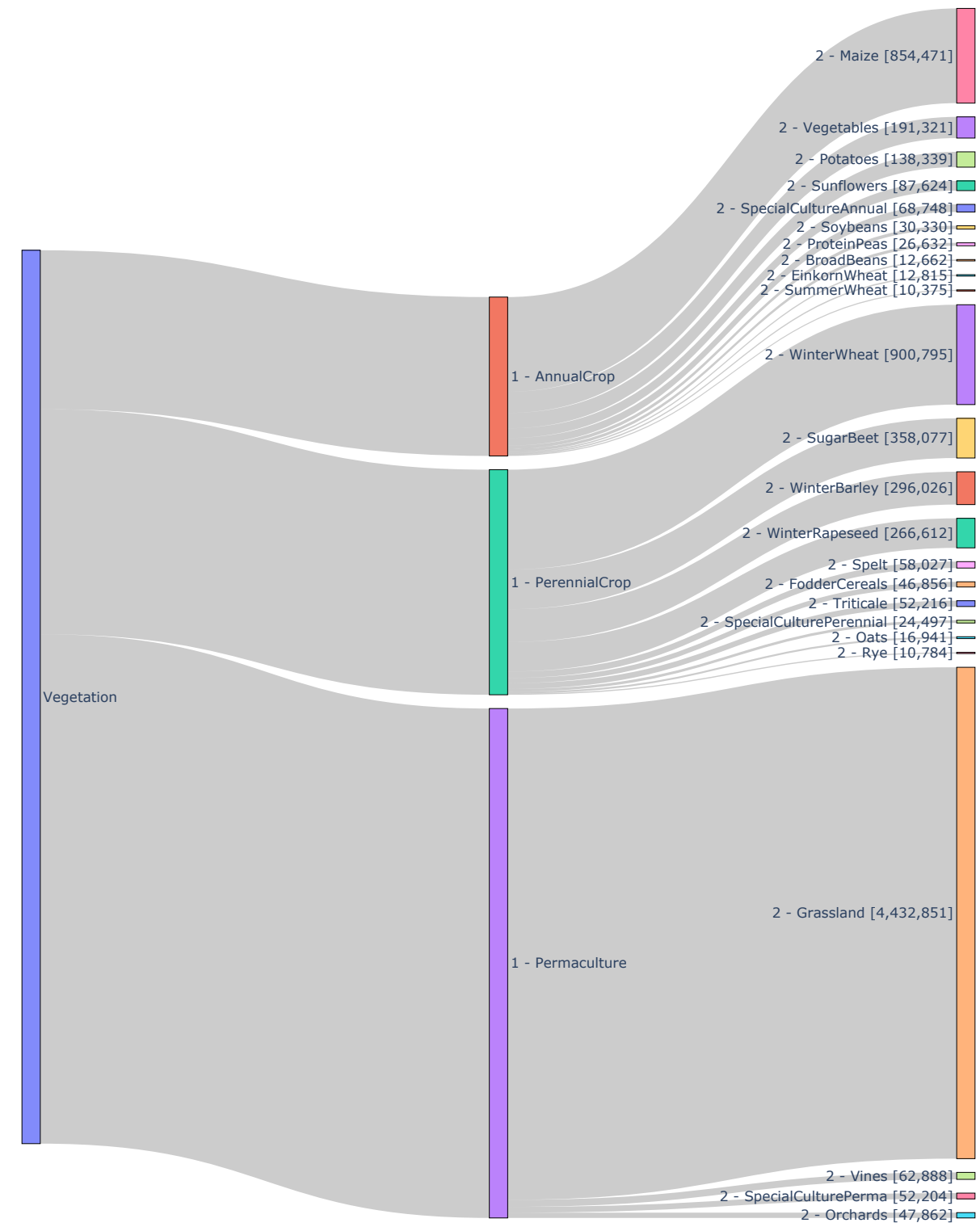



**Figure 5.4:** Sankey diagram of the ZueriCrop 2.0 dataset in the newly proposed Reduced Seasonality Hierarchy.

| Metric | † zueri_crop (5-fold) | seasonality (5-fold) | seasonality-reduced (5-fold) |
|---|---|---|---|
| F1 $T_1$ | 47.0% ± 7.4% | <u>82.4% ± 11.4%</u> | **82.6% ± 11.4%** |
| F1 $T_2$ | <u>30.5% ± 1.6%</u> | 23.7% ± 1.4% | **36.9% ± 0.5%** |
| F1 $T_3$ | 18.5% ± 1.3% | N/A | N/A |
| W. Acc. $T_1$ | 92.8% ± 0.3% | <u>93.5% ± 0.5%</u> | **93.6% ± 0.4%** |
| W. Acc. $T_2$ | 80.6% ± 1.8% | <u>85.4% ± 0.7%</u> | **86.2% ± 0.5%** |
| W. Acc. $T_3$ | 76.0% ± 1.4% | N/A | N/A |

**Table 5.7:** Results for experiment exp-7-label-hierarchy. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Insights:

1. **Baseline Performance (zueri_crop)**: The baseline setup using the original ZueriCrop 2.0 hierarchy performed similar to the baselines using this hierarchy in previous experiments.
2. **Seasonality-Based Hierarchy (seasonality)**: The new hierarchy resulted in a substantial performance improvement, especially in F1 scores. In Tier 1, the F1 score increased to 82.4% (+35.4pp) and weighted accuracy to 93.5% (+0.7pp). Note that the number of classes on this Tier in the hierarchy decreased from 5 to 3, compared to the baseline. For Tier 2, the F1 score decreased to 23.7% (-6.8pp) and weighted accuracy increased to 85.4% (+4.8pp). The seasonality hierarchy comprises 42 classes in Tier 2, in comparison to 16 in the baseline. Consequently, it is more appropriate to make comparisons with the baseline's Tier 3. The considerable increase in the number of classes can explain the lower F1 score in Tier 2. Notably, at the expense of 6 minority classes that are not captured by the seasonality label hierarchy, the seasonality hierarchy allows for substantial improvements in the remaining minority classes, as indicated by the +5.2pp increase of the F1 score in the most granular Tiers (Tier 3 of baseline and Tier 2 of seasonality hierarchy, respectively). However, the increased standard deviation in Tier 1 results suggests that training with the reduced hierarchy is less stable compared to the ZueriCrop 2.0 baseline.
3. **Reduced Seasonality-Based Hierarchy (seasonality_reduced)**: The reduced seasonality-based hierarchy, which includes 24 instead of 42 classes, shows even greater performance improvements, particularly in the most granular tier. At the expense of removing 18 more minority classes which the dataset only provided less than 10'000 pixels for, the F1 score increased by +13.2pp to 36.9% and the weighted accuracy by +0.8pp to 86.2% respectively.

Analysis of the model's confusion matrix (see Figure C.2 in appendix) reveals some newly introduced challenges with the seasonality hierarchy. The model has difficulty distinguishing between different seasonal variations of the same crop, such as Winter Wheat versus Summer Wheat and Winter Rapeseed versus Summer Rapeseed. As a result, crops with multiple seasonalities are often misclassified, with the model favoring the more prevalent seasonal variant - Winter in these cases. Overall, the results show that using a seasonality-based hierarchy significantly enhances model performance, at the cost of some granularity in the crop classification.

## 5.8 Dropping the Hierarchy

Here, we evaluate the impact of training without the hierarchical structure in the ZueriCrop 2.0 dataset as described in Section 3.2.1 Label Hierarchy. This is done by not computing a loss on the 3 hierarchical auxiliary heads, thus removing the hierarchical training. The hierarchy was created by Turkoglu et al. (2021) along with the hierarchical heads to address the low classification performance of minority classes in datasets with class imbalance such as ZueriCrop 2.0.

We wanted to test whether we can leverage this label hierarchy and the hierarchical auxiliary heads in our model architecture to improve the low classification performance of minority classes. The hypothesis is that the hierarchical structure, proven effective in the ms-convSTAR model, might not improve the F1 performance in our transformer-based architecture. The results can be observed in Table 5.8.

Experiment Setups:

1. **full_hierarchy (5-fold)**: This setup leverages all three tiers in the hierarchy by using three auxiliary heads during training to compute the loss and therefore derive the gradient (baseline).
2. **no_head (5-fold)**: This setup uses the same architecture as the other setup, except it computes no loss on the auxiliary heads.

| Metric | † full_hierarchy (5-fold) | no_head (5-fold) |
|---|---|---|
| F1 $T_1$ | 43.7% ± 3.5% | **43.8% ± 4.9%** |
| F1 $T_2$ | **30.7% ± 0.6%** | 30.3% ± 1.6% |
| F1 $T_3$ | 20.6% ± 0.8% | **20.8% ± 0.6%** |
| W. Acc. $T_1$ | **93.5% ± 0.2%** | 93.0% ± 0.4% |
| W. Acc. $T_2$ | **82.7% ± 0.3%** | 81.6% ± 1.1% |
| W. Acc. $T_3$ | **78.8% ± 0.6%** | 78.0% ± 1.4% |

**Table 5.8:** Results for experiment exp-8-drop-hierarchy. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Findings:

1. **full_hierarchy**: The baseline setup with the hierarchy achieved a slightly but significantly higher weighted accuracy for all Tiers (Tier 1: 93.5%, Tier 2: 82.7%, and Tier 3: 78.8%), and a marginally higher F1 on Tier 2 (30.7%) compared to the no_head setup, indicating a slight benefit in terms of accuracy when computing the loss for all hierarchical auxiliary heads.
2. **no_head**: This configuration performed slightly better in terms of F1 score for Tier 1 (43.8%) and Tier 3 (20.8%). However, this is not significant, as the increase of +0.1pp and +0.2pp respectively is well below 1 standard deviation.

These findings indicate that the full_hierarchy setup, which computes the loss on the hierarchical auxiliary heads, provides an improvement over the no_head configuration, as the full_hierarchy setup outperformed the no_head configuration in 3 out of 6 metrics significantly, but the no_head configuration did not significantly outperform the full_hierarchy setup in any reported metric. This suggests that the hierarchical structure and the hierarchical auxiliary heads do provide a significant but small advantage for our Prithvi architecture in addressing class imbalance.

## 5.9 Kernel Size in Hierarchical Head

In this experiment, we evaluate the impact of the kernel size used in the hierarchical head on the performance of our crop classification model. The hierarchical head design, as described in Section 4.1.3 Head: Spatially-Structured Embedding to Crop Classes, is inspired by the work of Jakubik et al. (2023), which employed a 3x3 kernel size in the conv2d layers. It seemed counterintuitive to use a 3x3 kernel since we expected no significant information to be extracted from neighboring pixels at this stage. To investigate whether including neighboring pixel information via the 3x3 kernel provides performance benefits, we compared it to a 1x1 kernel size configuration. The results can be obtained from Table 5.9.

Experiment Setups:

- **3x3 (5-fold)**: Utilizes a 3x3 kernel size in the conv2d layers within the hierarchical head (baseline).
- **1x1 (5-fold)**: Utilizes a 1x1 kernel size in the conv2d layers within the hierarchical head.

| Metric | † 3x3 (5-fold) | 1x1 (5-fold) |
|---|---|---|
| F1 $T_1$ | 47.0% ± 6.0% | **47.6% ± 4.4%** |
| F1 $T_2$ | 31.4% ± 1.5% | **32.1% ± 2.0%** |
| F1 $T_3$ | **20.3% ± 0.8%** | 19.7% ± 1.2% |
| W. Acc. $T_1$ | **93.2% ± 0.2%** | 93.0% ± 0.6% |
| W. Acc. $T_2$ | **82.0% ± 0.9%** | 81.8% ± 1.3% |
| W. Acc. $T_3$ | 77.2% ± 1.0% | **77.6% ± 1.7%** |

**Table 5.9:** Results for experiment exp-10-fcn-head-kernel-size. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Findings:

- **Baseline Performance (3x3)**: The 3x3 kernel setup achieved slightly higher weighted accuracy with a difference of 0.2pp for both Tier 1 (93.2%) and Tier 2 (82.0%) compared to the 1x1 kernel setup. The F1 score was marginally better for Tier 3 (20.3%) with a +0.6pp difference.
- **No neighboring pixel influence (1x1)**: The 1x1 kernel configuration outperformed the 3x3 kernel in F1 scores for Tier 1 (47.6%) and Tier 2 (32.1%) and had a higher weighted accuracy for Tier 3 (77.6%) with differences of +0.6pp, +0.7pp, and +0.4pp respectively, while using 6% fewer trainable parameters.

All metrics are within one standard deviation, except for the 3x3 kernel in the weighted accuracy for Tier 1. Even then, the score is only exactly one standard deviation better if judged by the 3x3 standard deviation. Therefore there is no significant performance difference when using a 3x3 kernel or a 1x1 kernel in the hierarchical head. However, the 1x1 kernel configuration uses 6% fewer trainable parameters, offering a more efficient model without sacrificing performance. In conclusion, the 1x1 kernel provides a computational advantage but comparable performance to the 3x3 kernel.

## 5.10 Parameter Efficiency in the Neck

In this experiment, we explore the impact of reducing the parameters of the neck module in Messis by using a ResNet bottleneck-like approach. The neck module is responsible for transforming the transformer tokens back to a spatially structured embedding with the same size as the input image before this is passed to the classification layers. The details of the neck are documented in Section 4.1.2 Neck: From Tokens to a Single Spatially-Structured Embedding. By introducing a bottleneck structure, we aim to reduce the computational complexity and number of parameters in the neck, while preserving essential features. This could improve model efficiency without sacrificing performance.

Each of the two modules in the baseline neck configuration consists of a sequence of two transposed convolutional layers. These layers output 2304 channels each (timesteps $\times$ embedding dimension $= 3 \times 768$) and include normalization and GELU activation functions between them. The bottleneck configuration of the neck introduces 1x1 convolutional layers before and after the transposed convolutional layers, which reduce and then restore the number of feature maps, thereby decreasing the overall parameter count in the neck. This approach is inspired by the concept of the bottleneck block in ResNet (K. He et al., 2015). In this new neck, we introduce the bottleneck reduction factor as a hyperparameter that reduces the original 2304 channels in the transposed convolutional layers by this factor.

Experiment Setups:

- **full-neck (5-fold)**: Standard neck configuration with the full set of parameters (baseline).
- **bottleneck-rf-4 (5-fold)**: Neck configuration with a reduction factor of 4, using a bottleneck structure. Reduces the number of trainable parameters in the neck from 84,953,088 to 10'641'024.

| Metric | † full-neck (5-fold) | bottleneck-rf-4 (5-fold) |
|---|---|---|
| F1 $T_1$ | 41.4% ± 4.6% | **47.3% ± 6.8%** |
| F1 $T_2$ | 28.8% ± 1.1% | **30.4% ± 2.3%** |
| F1 $T_3$ | **19.3% ± 1.0%** | 18.9% ± 0.5% |
| W. Acc. $T_1$ | **93.1% ± 0.3%** | 92.5% ± 0.4% |
| W. Acc. $T_2$ | **82.1% ± 0.9%** | 80.8% ± 1.5% |
| W. Acc. $T_3$ | **77.1% ± 1.3%** | 76.2% ± 1.3% |

**Table 5.10:** Results for experiment exp-13-neck-reduction. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Insights:

1. **Baseline Performance (full-neck)**: The baseline setup with the full set of parameters in the neck module achieved stronger performance in all scores but F1 scores on Tier 1 and Tier 2.
2. **Parameter Reduction (bottleneck-rf-4)**: The bottleneck configuration with a reduction factor of 4 showed an interesting trade-off. It outperformed the baseline in Tier 1 and Tier 2 F1 scores, achieving 47.3% (+5.9pp) and 30.4% (+1.6pp) respectively, suggesting that reducing the parameters can still maintain or even improve performance for higher tiers. However, for Tier 3 it slightly underperformed with an F1 score of 18.9% compared to 19.3% in the baseline. The weighted accuracies for the bottleneck-rf-4 configuration were slightly lower across all tiers compared to the baseline. Still, this setup demonstrated

that parameter reduction can enhance efficiency and performance in certain cases without a significant loss in lower-tier performance.

Overall, the results show that the reduction in parameters from 84,953,088 to 10,641,024 in the neck with setup bottleneck-rf-4 significantly decreases the computational complexity, making the model more efficient. This efficiency gain is achieved without substantial performance loss, especially in higher-tier classifications.

## 5.11 More Timesteps

One of the limitations of Prithvi is that it can only work with three timesteps at a time. Given that 142 images are available for analysis, this constraint will most likely result in the loss of significant temporal information. The goal of this experiment is to determine if increasing the number of timesteps fed into the model will enhance its performance. The workaround we chose involves creating multiple groups of three timesteps and processing each group through Prithvi. Prithvi's parameters are frozen and therefore all groups share the same weights. The outputs from Prithvi further pass through a separate neck for each group, after which the outputs get concatenated. This results in a multiplication of the number of embedding dimensions fed into the head by the number of groups. As the parameters from the neck make up more than 90% of the learnable parameters, this greatly increases the number of learnable parameters almost $n$-fold, where $n$ depicts the number of groups.

Experiment Setups:

- **3_timesteps (5-fold)**: Three images between 1st of May and 31st August 2019 with less than 5% cloud cover: 26.06.2019, 24.07.2019, and 30.08.2019 with time intervals of 28 and 37 days (baseline).
- **9_timesteps (5-fold)**: Nine images between 1st of March and 30th September 2019 with less than 5% cloud cover selected evenly throughout the time range: 21.03.2019, 20.04.2019, 01.06.2019, 04.06.2019, 26.06.2019, 24.07.2019, 18.08.2019, 04.09.2019, and 29.09.2019, with time intervals of 30, 42, 3, 22, 28, 25, 17, and 25 days.

| Metric | † 3_timesteps (5-fold) | 9_timesteps (5-fold) |
|---|---|---|
| F1 $T_1$ | 47.8% ± 5.2% | **50.1% ± 3.7%** |
| F1 $T_2$ | 31.5% ± 1.5% | **32.5% ± 1.0%** |
| F1 $T_3$ | 19.5% ± 1.6% | **21.4% ± 0.9%** |
| W. Acc. $T_1$ | 93.0% ± 0.2% | **93.5% ± 0.2%** |
| W. Acc. $T_2$ | 81.2% ± 1.0% | **82.6% ± 0.7%** |
| W. Acc. $T_3$ | 76.7% ± 1.0% | **77.0% ± 1.1%** |

**Table 5.11:** Results for experiment exp-14-more-timesteps. All metrics calculated on field majority during validation phase, with mean and std reported for each tier $T_n$. Best scores bold, second best underlined, † is baseline.

Key Findings:

- **More Timesteps (9_timesteps)**: The setup using nine timesteps showed improved performance across all metrics. It achieved F1 scores of 50.1% for $T_1$, 32.5% for $T_2$, and 21.4% for $T_3$ with a difference of +2.3pp, +1.0pp, and +1.9pp. The weighted accuracy scores were 93.5% for $T_1$, 82.6% for $T_2$, and 77.0% for $T_3$ with a difference of +0.5pp, +1.4pp, and +0.3pp.

The results indicate that increasing the number of timesteps enhances the performance of the model. The nine-timesteps configuration outperformed the three-timesteps baseline in all metrics, with significant improvements in the F1 score for $T_3$ and the weighted accuracy for $T_1$ and $T_2$ of more than 1 standard deviation. The approach of using multiple groups of timesteps to bypass the model's three-timestep limitation used close to two times the hours to train as seen in Figure 5.5. Nevertheless, it proves effective, yielding better classification accuracy and F1 scores but not close to the metrics reported by Turkoglu et al. (2021).

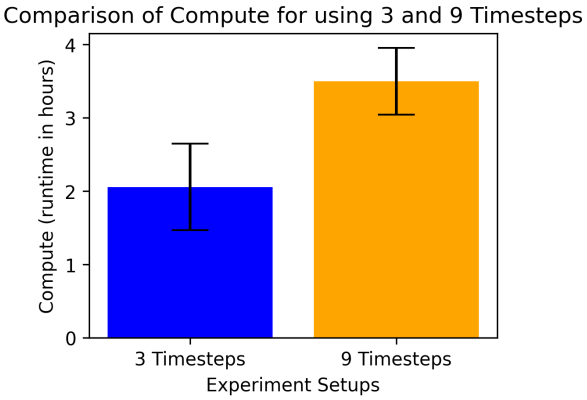



**Figure 5.5:** Comparison of computing time required to train Messis with 3 vs 9 timesteps. Results are averaged and the error bars report standard deviation across the 5 folds.

## 5.12 Best Model: Messis

Based on our extensive experimentation, we combined the adaptations that led to significant performance gains and developed an enhanced version of Messis - our best model. We compare this version of Messis with the results reported for the best ms-ConvSTAR model by Turkoglu et al. (2021). The performance differences are analyzed to address our research questions. Results from this evaluation, conducted on the sixth fold — which was reserved exclusively for testing and not used in any hyperparameter tuning or model adjustments — are presented in Table 5.13.

Model Setup:

| Experiment | Chosen Value |
|---|---|
| Tuning the Learning Rate | 1e-4 |
| Impact of Dropout Type and Probability | Dropout2D with probability 0.1 |
| Dataset Fold Split Strategy | Stratified |
| Comparing Optimizers | Adam without weight decay |
| Satellite Imagery Time Ranges | April - August |
| Fine-tuning Prithvi | unfrozen |
| Improving Generalization with Data Augmentation | Flip Probability = 0.5, Jitter Std. = 0.1 |
| Ignoring the Background in Loss Calculation | ignore |
| Alternative Label Hierarchy | Remote-sensing focused ZueriCrop |
| Dropping the Hierarchy | Full hierarchy |
| Kernel Size in Hierarchical Head | 1x1 |
| Parameter Efficiency in the Neck | Full neck |
| More Timesteps | 9 |
| More Convolutional Layers in Hierarchical Heads | 1 |
| More Channels in Hierarchical Heads | 1024 |

**Table 5.12:** The model architecture choices and hyperparameters used for the best Messis model.

Key Insights:

1. **Tier 1 Performance**: The best Messis model demonstrated strong performance in Tier 1 metrics, significantly outperforming ms-ConvSTAR in F1 score and precision and recall. Specifically, it achieved an F1 score of 64.0% (+7.0pp), a precision of 83.5% (+3.2pp), and a recall of 57.6% (+4.7pp) on the test fold, indicating that Messis was particularly effective at Tier 1. However, the weighted accuracy was slightly lower than ms-ConvSTAR's, with a score of 95.5% compared to 96.3%.
2. **Tier 2 and Tier 3 Performance**: While ms-ConvSTAR maintains relatively steady metrics across all three tiers, with only slight decreases as the number of classes to differentiate increases, Messis exhibits a significant drop in performance from Tier 1 to Tier 2, particularly in precision and recall. Specifically, Messis sees a drop in precision from 83.5% in Tier 1 to 50.6% in Tier 2 (-32.9pp) and in recall from 57.6% to 37.4% (-20.2pp). This sharp decline in precision and recall leads to a corresponding decrease in the F1 score for Tier 2 (40.1%, down from 64.0% in Tier 1). Although the drop from Tier 2 to Tier 3 is less pronounced, it is still more significant than the decline observed in ms-ConvSTAR, with the F1 score in Tier 3 falling to 34.8%. This suggests that Messis struggles more with the increasing complexity and number of classes at each tier (6 classes in Tier 1, 16 in Tier 2, and 48 in Tier 3).

| Metric | Messis [val] (5-fold) | Messis [test] (5-fold) | ms-convSTAR (5-fold) |
|---|---|---|---|
| F1 $T_1$ | 62.7% ± 7.3% | **64.0% ± 2.3%** | 57.0% |
| F1 $T_2$ | 42.3% ± 3.7% | 40.1% ± 1.5% | **54.5%** |
| F1 $T_3$ | 35.9% ± 0.6% | 34.8% ± 1.4% | **52.4%** |
| W. Acc. $T_1$ | 95.6% ± 0.2% | 95.5% ± 0.1% | **96.3%** |
| W. Acc. $T_2$ | 86.3% ± 0.5% | 85.6% ± 0.4% | **89.2%** |
| W. Acc. $T_3$ | 83.6% ± 0.7% | 82.9% ± 0.5% | **88.0%** |
| Precision $T_1$ | 78.6% ± 5.4% | **83.5% ± 2.9%** | 80.3% |
| Precision $T_2$ | 55.4% ± 4.3% | 50.6% ± 1.5% | **73.0%** |
| Precision $T_3$ | 46.6% ± 1.6% | 44.2% ± 3.6% | **60.1%** |
| Recall $T_1$ | 57.6% ± 7.6% | **57.6% ± 2.3%** | 52.9% |
| Recall $T_2$ | 39.0% ± 3.6% | 37.4% ± 1.5% | **51.0%** |
| Recall $T_3$ | 32.9% ± 0.7% | 31.7% ± 0.9% | **49.8%** |
| Kappa $T_1$ | 0.914 ± 0.004 | 0.912 ± 0.002 | N/A |
| Kappa $T_2$ | 0.803 ± 0.007 | 0.794 ± 0.003 | N/A |
| Kappa $T_3$ | 0.773 ± 0.01 | 0.764 ± 0.004 | N/A |

**Table 5.13:** Results for experiment best-model. Messis metrics calculated on field majority during validation and test phase, with mean and std reported for each tier $T_n$. Best scores bold. Metrics for ms-ConvSTAR are derived from the validation fold, as reported by Turkoglu et al. (2021) in Table 4.

3. **Cohen's Kappa**: The cohen's kappa values for Tier 1, Tier 2, and Tier 3 are 0.912, 0.794, and 0.764, respectively. These values indicate a high level of agreement beyond chance, with Tier 1 showing almost perfect agreement, and Tiers 2 and 3 reflecting substantial agreement.

To further explore this drop in performance and the differences in classification behavior between Messis and ms-ConvSTAR, it is helpful to examine the detailed per-class accuracy for both models across the tiers shown in Table 5.14. Notably, Messis outperformed ms-ConvSTAR in identifying certain classes. For example, across all tiers, Messis achieved an accuracy of 30.97% for the Forest class, compared to just 1.6% for ms-ConvSTAR. Similarly, for Special Crops in Tier 1, Messis outperformed ms-ConvSTAR by 5.37% in accuracy. These two factors contribute significantly to the lead Messis has in Tier 1 performance.

A comparison of per-class accuracy in Tier 3 reveals that Messis outperforms ms-ConvSTAR in detecting Forest, Pears and Hedge. For all other classes, Messis performs worse, with accuracy differences ranging from -1.13pp (Maize) to -56.63pp (Legumes). When focusing on the top 5 majority classes Meadow, Winter Wheat, Maize, Pasture, and Sugar Beets, Messis consistently lags behind ms-ConvSTAR. The performance differences are generally small for Meadow (-2.21pp), Winter Wheat (-2.33pp), Maize (-1.13pp), and Sugar Beets (-0.54pp). However, for Pasture, Messis experiences a significant drop in accuracy, underperforming by -17.06pp. The confusion matrix for Tier 3 (Figure C.2) reveals that this drop is primarily due to the model's inability to accurately distinguish Pasture from Meadow. The classes Wheat, Spelt, Oat, EinkornWheat, SummerWheat, and Buckwheat are frequently misclassified as Winter Wheat. Chicory and Pumpkin are often misclassified as Vegetables, and all instances of SummerRapeseed are misclassified as WinterRapeseed. Thus, the differentiation of crops with similar appearances is a common challenge for the model.

| Tier 1 | Tier 1 Accuracy | Tier 2 | Tier 2 Accuracy | Tier 3 | Tier 3 Accuracy |
|---|---|---|---|---|---|
| Field crops (42.86%) [-1.46] | 96.74 ± 0.37 | BroadLeafRowCrop (11.6%) [-6.74] | 88.36 ± 1.58 | Beets (0.01%) [0.00] | 0.00 ± 0.00 |
| | | | | Field bean (0.16%) [-16.61] | 71.69 ± 1.93 |
| | | | | Hemp (0.03%) [-3.10] | 0.00 ± 0.00 |
| | | | | Legumes (0.05%) [-56.63] | 21.37 ± 32.47 |
| | | | | Linen (0.05%) [-21.80] | 32.60 ± 31.85 |
| | | | | Lupine (0.0%) [0.00] | 0.00 ± 0.00 |
| | | | | Mustard (0.0%) [N/A] | N/A |
| | | | | Peas (0.33%) [-15.83] | 66.57 ± 13.69 |
| | | | | Potatoes (1.72%) [-20.61] | 65.99 ± 2.91 |
| | | | | Soy (0.38%) [-37.15] | 50.25 ± 5.33 |
| | | | | Sugar_beets (4.43%) [-0.54] | 96.66 ± 0.45 |
| | | | | SummerRapeseed (0.02%) [0.00] | 0.00 ± 0.00 |
| | | | | Sunflowers (1.09%) [-8.40] | 84.40 ± 1.37 |
| | | | | Tobacco (0.03%) [-48.50] | 0.00 ± 0.00 |
| | | | | WinterRapeseed (3.31%) [-3.49] | 94.11 ± 1.45 |
| | | CropMix (0.09%) [-16.70] | 0.00 ± 0.00 | MixedCrop (0.09%) [-39.70] | 0.00 ± 0.00 |
| | | LargeGrainCereal (10.68%) [-1.16] | 93.14 ± 0.85 | Maize (10.6%) [-1.13] | 93.57 ± 0.76 |
| | | | | Sorghum (0.07%) [-27.00] | 0.00 ± 0.00 |
| | | SmallGrainCereal (17.56%) [-0.54] | 97.16 ± 0.33 | Buckwheat (0.02%) [-7.00] | 0.00 ± 0.00 |
| | | | | EinkornWheat (0.16%) [-33.62] | 16.18 ± 9.70 |
| | | | | Grain (0.04%) [-21.70] | 0.00 ± 0.00 |
| | | | | Oat (0.21%) [-50.25] | 26.15 ± 14.94 |
| | | | | Rye (0.13%) [-47.68] | 20.42 ± 7.10 |
| | | | | Spelt (0.72%) [-31.76] | 52.24 ± 10.59 |
| | | | | SummerBarley (0.06%) [-19.10] | 7.00 ± 15.66 |
| | | | | SummerWheat (0.13%) [-13.79] | 24.21 ± 4.71 |
| | | | | Wheat (1.23%) [-13.24] | 26.16 ± 8.37 |
| | | | | WinterBarley (3.67%) [-2.57] | 93.83 ± 1.08 |
| | | | | WinterWheat (11.18%) [-2.33] | 93.37 ± 1.84 |
| | | VegetableCrop (2.93%) [-7.21] | 76.19 ± 5.03 | Chicory (0.08%) [-6.69] | 24.11 ± 11.79 |
| | | | | Pumpkin (0.03%) [-14.20] | 0.00 ± 0.00 |
| | | | | Vegetables (2.83%) [-5.68] | 76.62 ± 4.98 |
| Forest (0.16%) [+29.37] | 30.97 ± 13.51 | Forest (0.16%) [+29.37] | 30.97 ± 13.51 | Forest (0.16%) [+29.37] | 30.97 ± 13.51 |
| Grassland (54.47%) [-0.41] | 97.49 ± 0.25 | BiodiversityArea (0.05%) [0.00] | 0.00 ± 0.00 | Biodiversity enc. area (0.05%) [0.00] | 0.00 ± 0.00 |
| | | Meadow (46.85%) [-2.61] | 92.69 ± 3.02 | Meadow (46.85%) [-2.21] | 92.69 ± 3.02 |
| | | Pasture (7.58%) [-12.06] | 25.34 ± 12.76 | Pasture (7.58%) [-17.06] | 25.34 ± 12.76 |
| Orchards (1.51%) [-0.58] | 49.02 ± 5.17 | OrchardCrop (1.37%) [-25.26] | 51.04 ± 5.18 | Apples (0.41%) [-22.50] | 46.80 ± 3.65 |
| | | | | Chestnut (0.0%) [N/A] | N/A |
| | | | | Hops (0.01%) [0.00] | 0.00 ± 0.00 |
| | | | | Pears (0.08%) [+18.20] | 20.50 ± 14.28 |
| | | | | StoneFruit (0.09%) [-16.40] | 7.70 ± 8.93 |
| | | | | Vines (0.78%) [-30.39] | 51.81 ± 5.13 |
| | | TreeCrop (0.14%) [-12.99] | 7.91 ± 3.39 | TreeCrop (0.14%) [-29.29] | 7.91 ± 3.39 |
| Special crops (1.0%) [+5.37] | 13.67 ± 5.75 | Berries (0.21%) [-6.47] | 12.33 ± 10.80 | Berries (0.21%) [-13.47] | 12.33 ± 10.80 |
| | | Fallow (0.37%) [-44.08] | 21.72 ± 9.98 | Fallow (0.37%) [-48.58] | 21.72 ± 9.98 |
| | | Gardens (0.0%) [-1.00] | 0.00 ± 0.00 | Gardens (0.0%) [0.00] | 0.00 ± 0.00 |
| | | Hedge (0.41%) [+1.05] | 2.05 ± 1.60 | Hedge (0.41%) [+0.55] | 2.05 ± 1.60 |
| | | Multiple (0.01%) [-1.00] | 0.00 ± 0.00 | Multiple (0.01%) [0.00] | 0.00 ± 0.00 |

**Table 5.14:** Per-class accuracy/recall for experiment best-model. Messis metrics calculated on field majority during test phase over 5 folds, with mean and std reported for each tier $T_n$. Class frequencies in ZueriCrop 2.0 dataset are given in parenthesis. Performance difference to ms-ConvSTAR in percentage points given in square brackets. Per-class accuracy for ms-ConvSTAR is derived from table C.8 in the work of Turkoglu et al. (2021).

# 6 Discussion

## 6.1 RQ 1: Adapting Prithvi to the ZueriCrop Dataset

The first main research question we aimed to answer was: "Can the geospatial foundation model Prithvi be adapted (fine-tuned) to the ZueriCrop dataset?"

The results achieved with Messis in Experiment 5.12 Best Model: Messis demonstrate that Prithvi is not only a viable but also a suitable choice for developing a crop classification model for Switzerland. While Prithvi was pre-trained on HLS satellite imagery from diverse regions across the United States, with a pixel resolution of 30x30 meters (Jakubik et al., 2023), Messis employs our ZueriCrop 2.0 dataset, which was generated from Sentinel-2 satellite imagery of the Swiss cantons of Zürich and Thurgau. This dataset features smaller field sizes and a higher pixel resolution of 10x10 meters. Despite these fundamental differences — specifically, the pretraining data's origin from the American continent, the smaller field sizes, and the higher pixel resolution — Prithvi's performance was not hindered. This adaptability of Prithvi to unseen regions and higher resolutions was also demonstrated by Jakubik et al. (2023) in their global flood mapping application, which aligns with our findings.

Experiment 5.4 Fine-tuning Prithvi comparing a transfer learned, fine-tuned, and non-pretrained variant of Prithvi clearly showed that fine-tuning can enhance performance over mere transfer learning. Moreover, using the pre-trained Prithvi as the backbone of Messis resulted in significantly better performance compared to a model variant trained from scratch without pretraining, as shown in Experiment 5.4 Fine-tuning Prithvi. Therefore, our findings provide further evidence of Prithvi's capacity to encode essential features of geospatial data and establish it as a universal model that can adapt to a variety of applications.

Further, we sought to answer the sub-question "What specific challenges are associated with fine-tuning Prithvi, and what methods could potentially overcome them?". One central challenge presented itself in the context of the label hierarchy. In contrast to a simple, single-head classification, a hierarchy introduces additional complexity. It is a way of nudging the model to learn in a hierarchical manner, with more detailed predictions up until the most granular level being established over several tiers. Our approach, combining several hierarchical heads in sequence followed by a refinement head, with a separate loss computed on each head, allows for this, as described in Section 4.1.3 Head: Spatially-Structured Embedding to Crop Classes. The experiments demonstrated that Messis effectively leverages Prithvi's upscaled embeddings provided by the model neck to classify pixels within the given hierarchy. The challenge of cloud presence in satellite imagery can be circumvented by establishing a maximum threshold for cloud coverage in the satellite imagery used for producing the ZueriCrop 2.0 dataset. However, if the time range considered for producing the dataset is too narrow, this may result in insufficient data due to the high likelihood of cloud-covered images. Additionally, it may not fully capture the relevant crop growth activity phases, as observed in Experiment 5.3 Satellite Imagery Time Ranges. Being somewhat resource-constrained, we experimented with several ways of reducing computational complexity. Experiment 5.10 Parameter Efficiency in the Neck demonstrated that a ResNet-inspired bottleneck approach for the neck of Messis helped reduce the neck's parameter count by a factor of around 8. The utilization of 1x1 kernels instead of 3x3 in the hierarchical head, as examined in Experiment 5.9 Kernel Size in Hierarchical Head, further contributed to this reduction.

One fundamental incompatibility that needed to be solved was the dimension of the input satellite images. While the original ZueriCrop dataset provides 24x24 pixels, Prithvi uses an image size of 224x224 pixels. Since stitching together patches to increase the dimension was not feasible due to the lack of geographical references in the ZueriCrop dataset, we opted to

build a DVC pipeline. This pipeline reconstructs the dataset from scratch using Sentinel-2 satellite imagery and the ground truth data provided by the cantons, as explained in Section 3.3 ZueriCrop Dataset. This allowed us to explore another possible area of improvement we identified in the ZueriCrop datset, the fold split by band. Experiment 5.1 Dataset Fold Split Strategy exploring different fold split strategies showed that, contrary to our expectations, a stratified split strategy does not significantly improve model performance. One drawback of Prithvi's larger chip size is that, on consumer-grade GPUs, it limits the scalability of the number of timesteps that can be utilized, particularly when following the approach used in Experiment 5.11 More Timesteps. This is due to a higher VRAM usage as the model size and specifically the number of necks increase with each timestep.

Our results confirm that Prithvi can be fine-tuned to the ZueriCrop dataset, successfully addressing the challenges related to resolution, label hierarchy, and computational complexity. We have demonstrated that Prithvi's adaptability makes it a suitable choice for crop classification tasks in diverse geographic regions. These findings reinforce Prithvi's potential as a versatile geospatial foundation model, capable of adapting to new datasets with varying characteristics.

## 6.2 RQ 2: How does Messis compare against ms-ConvSTAR?

Our second main research question aims to answer: "How does the performance of the fine-tuned Prithvi model in crop classification on the ZueriCrop dataset compare to the existing models of Turkoglu et al. (2021)?"

The comparative analysis in Experiment 5.12 Best Model: Messis reveals that while Messis demonstrates considerable potential, it does not fully reach the performance levels of the ms-ConvSTAR model developed by Turkoglu et al. (2021). Although Messis outperforms ms-ConvSTAR in Tier 1, particularly in precision and recall, it experiences significant drops in performance as the complexity of the classification task increases in Tiers 2 and 3. This suggests that while the Prithvi model, as fine-tuned within Messis, is highly effective for simpler classification tasks, there remain challenges in handling more complex, hierarchical classifications with a larger number of classes.

In answering the sub-question "What quantitative metrics can be applied to effectively evaluate the performance differences?", we utilized F1 macro and weighted accuracy as primary metrics for comparison. Additionally, we examined confusion matrices and the per-class accuracy reported by Turkoglu et al. (2021) against the per-class accuracy obtained with Messis. The results show that Messis underperforms in most per-class metrics, with varying degrees of severity. This may be attributed to factors such as:

- The fewer number of timesteps (9) used in Messis compared to ms-ConvSTAR, which utilized a total of 71 timesteps.
- The increased number of satellite bands (6 instead of 4), including two additional bands with a lower resolution of 20x20 meters, might affect classification accuracy. Turkoglu et al. (2021) has shown that using 9 satellite bands instead of 4 degrades ms-ConvSTAR performance.
- Differences in the architectural approach to learning hierarchies. Unlike ms-ConvSTAR, proposed by Turkoglu et al. (2021), which uses a hierarchical architecture with multiple layers of convolutional recurrent neural networks, specifically ConvSTAR cells, Messis leverages the Prithvi Vision Transformer (ViT) as its backbone. In their model, ConvSTAR cells are employed in each hierarchical tier to process spatio-temporal features sequentially, followed by a conventional CNN classifier. This approach allows the model to refine its predictions for each subsequent tier through a series of recurrent and convolutional operations. In contrast, Messis utilizes Prithvi's ViT, with the attention mechanisms handling

feature extraction and hierarchical learning, before passing the outputs through a neck for upsampling. Messis then uses convolutional heads for the final classification, without any sequential processing layers between hierarchical levels. This potentially impacts the performance of Messis and its ability to leverage hierarchical information to some degree.

Regarding the sub-question "Does changing the label hierarchy from a remote-sensing focused hierarchy (Turkoglu et al., 2021 / ZueriCrop) to a seasonality-based hierarchy positively impact the model performance?", our findings suggest that the seasonality-based label hierarchy does have a beneficial effect on overall model performance. However, reducing the number of classes in the hierarchy also simplifies the classification problem, which can improve scores. It remains unclear whether the improvements are due to grouping by seasonality or simply the reduction in the number of classes. Experiment 5.8 Dropping the Hierarchy answers whether the hierarchy is advantageous at all. We found that the label hierarchy improves model performance, supporting Turkoglu et al.'s (2021) findings that the hierarchical approach performs superior. Still, we did not see as big a change in performance as reported by them (+9.9pp on F1 score). This could be due to the different approach they took by removing the refinement head when eliminating the hierarchy. The fact that crop classification on coarser levels of the hierarchy is more accurate, as we also found with Messis, makes the hierarchy helpful for applications like summary statistics or calculation of subsidies which do not require the most granular detail (Turkoglu et al., 2021).

In summary, our findings indicate that while Messis shows promise, it falls short of matching the performance of ms-ConvSTAR. Messis underperforms in most per-class metrics, potentially due to fewer timesteps, the inclusion of additional lower-resolution satellite bands, and architectural differences in hierarchical learning approaches. Furthermore, while the seasonality-based label hierarchy appears to enhance overall performance, it is unclear whether this improvement is due to the hierarchy itself or simply the reduced number of classes.

## 6.3 RQ 3: The effect of higher temporal resolution

Our third main research question aimed to address: "Does the use of higher temporal resolution data have a positive effect on model performance?"

The results indicate that increasing the number of timesteps does enhance model performance significantly. Utilizing nine timesteps instead of three shows a notable improvement in accuracy as shown in Experiment 5.11 More Timesteps. This increase in performance can likely be attributed to the richer temporal information provided by more frequent observations, which helps the model capture various stages of crop planting, growth, harvesting, and seasonal changes that were not captured with the reduced frequency that three timesteps provide.

To address the sub-question, "What are the implications of a higher temporal resolution when using Prithvi, which was pre-trained on 3 timesteps?", our findings indicate several key points:

While there is a one-to-two percentage point increase in performance, depending on the metric, more timesteps do not completely alter the results. This marginal performance improvement is accompanied by a significant increase in the number of parameters and VRAM usage. With an additional two necks for each group of 3 timesteps required, each of them 84 953 088 parameters in size, the total parameters are increased by 169 906 176. With the previous model size of Messis being 178 462 329, this almost doubles (+95.20%) the number of parameters. We did not conduct experiments with more than nine timesteps, as we believe that further increasing the timesteps in this manner would not be a worthwhile pursuit. Instead, alternative approaches could be explored, such as using a single neck and concatenating the outputs at the embedding dimension after the backbone, rather than employing multiple necks. Additionally, the resulting embedding dimensions can be summed or averaged instead of employing multiple necks.

Moreover, while additional timesteps offer richer data, they potentially introduce challenges. For instance, using more frequent satellite images will increase the likelihood of encountering images with cloud coverage exceeding our maximum threshold of 5%. In light of the demonstrated capabilities of Prithvi demonstrated in Cloud Gap Imputation by Jakubik et al. (2023), it is reasonable to expect that the feature extraction will continue to function effectively, thereby preventing a significant degradation in the performance of Messis. Nevertheless, this potential issue requires further investigation and testing to fully assess its impact.

In addition to the number of timesteps, the chosen timespan is a significant factor. We conducted Experiment 5.3 Satellite Imagery Time Ranges to explore the impact of varying time ranges. The results demonstrated that the period from April to August significantly outperformed the alternative periods from March to September and May to August. This illustrates that the selected timespan can be either too brief or too lengthy and that a good balance needs to be identified. It is crucial to include images with such frequency that they are able to cover all the relevant crop activities. It would be interesting to explore how the seasonality-based hierarchy performs with more timesteps, as this could support the model's ability to learn and distinguish seasonal patterns more effectively. Given the diversity of crops in our dataset, identifying the optimal time period and sample frequency is challenging. Expert knowledge in the domain of crop science could potentially enhance performance by providing insights on the level of specific crops for more precise timing and sampling strategies.

In summary, our findings revealed that while increasing the temporal resolution by using more timesteps does lead to a measurable improvement in model performance, the benefits come with trade-offs in terms of the number of parameters and therefore VRAM usage. Alternative model architecture approaches to handle an even bigger number of timesteps may be promising. Additionally, the results highlighted the importance of selecting an appropriate timespan for data collection, as the choice of time range significantly impacted model accuracy.

## 6.4 Additional Contributions to the Field of Crop Classification

We conducted six experiments that were not directly related to the Research Questions answered above. The following paragraphs summarize and interpret them, highlight strengths and limitations, compare them to theory, and explore potential applications.

Our own strategy to split data into stratified folds for multi-label data with counts (stratified strategy) outperformed a random strategy and matched the performance of methods by Turkoglu et al. (2021) (band strategy) and a method from Sechidis et al. (2011) expanded to deal with multi-label data with counts (sechidis strategy), as shown in Figure 6.1. It was challenging to stratify our heavily skewed long-tail multi-label distribution dataset, as it had counts per sample and class. It was even more difficult since it had very few samples for minority classes. Nevertheless, our method showed promise and was able to keep up with the band and sechidis strategy. Our approach was only evaluated on one dataset, indicating a need for testing across multiple multi-label distribution datasets with counts in different domains to fully assess its usefulness and robustness.

When comparing 4 different optimizers to train our model, we observed the superiority of AdamW over SGD as reported by Kumar et al. (2023). We also expected to observe an advantage of AdamW over Adam as mentioned in other studies (Llugsi et al., 2021; Loshchilov & Hutter, 2019; Zhou et al., 2024), but failed to do so. This outcome suggests potential limitations related to the learning rate and weight decay hyperparameter used. Further experimentation with different learning rates and weight decay parameters might yield better performance from AdamW. But we also cannot rule out the possibility that Adam is just superior over AdamW in our task.

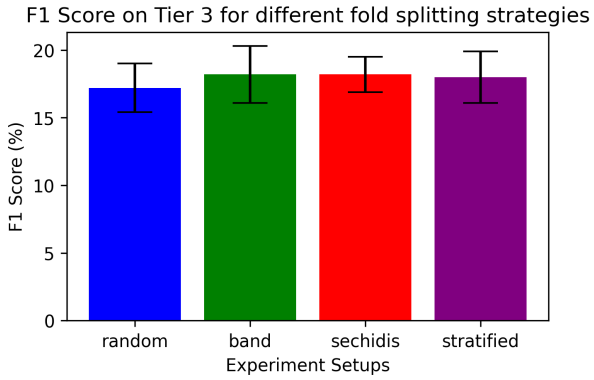



**Figure 6.1:** Comparison of the four different fold splitting strategies based on the F1 score on Tier 3. Results are averaged and the error bars report standard deviation across the 5 folds.

Data augmentation improved our results significantly by partially mitigating the negative effects of overfitting on our small training dataset. This is in line with observations from other research, supporting the fact that data augmentation improves the performance of CNN models in remote sensing tasks (Dave & Pandya, 2020; Xingrui Yu & Ren, 2017). While we did only explore two data augmentation techniques, incorporating additional methods (e.g. rotation, contrast, blur, illumination) could potentially enhance the achievable model performance with our small training dataset further (Pawara et al., 2017).

Ignoring the background class in the loss calculation led to an improved model performance, likely by partially alleviating issues related to class imbalance. Excluding the largest class Background, which constituted about 70% of the dataset, allowed the model to focus more on the crops. However, this approach has limitations, particularly when the bounds of fields are unknown, as the model demonstrated a weakened ability to identify field boundaries compared to the version trained considering background loss. This limitation is not problematic, as our model is tailored for crop classification rather than segmentation tasks. Therefore, it was not developed with the intent to identify field boundaries or perform segmentation. Although the results suggest potential for segmentation applications, especially if the model were designed with this use case in mind from the outset, such capabilities are beyond the current scope of our work.

Experiment 5.9 Kernel Size in Hierarchical Head indicated that the hierarchical heads did not extract information from neighboring pixels, suggesting that the Prithvi backbone and neck were already effective in capturing necessary details. This can be attributed to the neck's capability to inflate the 14x14x3 embeddings back to the original image dimensions of 224x224x3, ensuring information propagation to the correct pixel.

We demonstrated that it is possible to significantly reduce the number of parameters by a factor of 8 in the neck without compromising performance, using techniques similar to the residual block in ResNet (K. He et al., 2015). Even though we reduced the number of trainable parameters outside of the transformer part, this aligns with findings in pretrained transformer models, where minimal parameter adaptation is required during the adaption/fine-tuning (X. He et al., 2023; Xin et al., 2024).

The combined insights from these experiments provide several key contributions to the field of crop classification. Our custom dataset fold split strategy, though requiring further validation, offers a potential improvement for handling long-tail distributions. The limited benefit of AdamW in Experiment 5.2 Comparing Optimizers suggests the importance of optimizer-specific

tuning. Data augmentation remains a robust method for enhancing generalization, and ignoring background in loss calculation can effectively address parts of class imbalance. Architectural modifications show promise in maintaining performance with fewer parameters, contributing to more efficient model designs.

## 6.5 Potential Applications for Messis

The advancements made in our work have significant potential applications, particularly in the field of crop classification. While the performance of Messis has not surpassed that of the ZueriCrop ms-ConvSTAR model by Turkoglu et al. (2021), our experiments demonstrated that geospatial foundation models, like Prithvi, are well-suited for fine-tuning tasks where data availability is limited and can effectively handle crop classification.

Our model Messis performed well for majority classes, which are widely grown in Switzerland, especially in regions like Zurich (ZH) and Thurgau (TG). This can be observed in Table 5.14 where 6 out of the 11 crops with more than a 1% share in the ZueriCrop 2.0 dataset achieved a Tier 3 accuracy of above 90% (*Sugar beets, WinterRapeseed, WinterBarley, Maize, WinterWheat,* and *Meadow*), three more (*Sunflowers, Vegetables,* and *Potatoes*) scored above 65%, and the remaining two crops (*Wheat* and *Pasture*) achieving an accuracy of only around 25%. The crops Meadow, Maize, and WinterWheat, which Messis classifies really well, are the economically relevant crops for Swiss agriculture. However, Messis struggled to learn minority classes due to the limited data available for these crops and our best model did not surpass the ZueriCrop ms-ConvSTAR model by Turkoglu et al. (2021) on Tier 2 and Tier 3. This presents a challenge when specific crops need to be identified at a tier 3 granularity, where detailed classification is required. This enhances the model's utility for statistical aggregations where data precision at the most granular level isn't required. However, it doesn't substitute the manual task of drawing field polygons and documenting crop types, which would reduce the administrative burden on Swiss farmers doing this every year.

# 7 Future Work

Our experiments reveal promising potential for extending the current work into segmentation tasks. Specifically, Experiment 5.6 Ignoring the Background in Loss Calculation indicates that Messis can be adapted to segment fields without relying on pre-defined ground-truth polygons. Future work should address the currently arbitrary nature of the background class, for example by adding ground truth for real background areas like infrastructure, thus allowing the model to differentiate between field and non-field areas. This additional data could be sourced from the swissTLM3D dataset (Bundesamt für Landestopografie swisstopo, 2024). If the model generalizes well as a result, in a next step, an innovative approach could involve self-supervised learning. The model would be employed to generate new ground-truth data for pixels previously wrongly marked as background, for example forest areas. This newly created dataset would then enhance the model's performance and generalization. Making this an iterative process could enhance performance, leading to a model capable of accurately segmenting and classifying fields in any satellite image. This approach would enable the generation of crop statistics for a specific region using only satellite images, without necessitating field polygons.

Another potential direction for future work involves defining a new label hierarchy based on where the model struggles, especially if there's no immediate requirement to differentiate between slight variations of crops like Winter Wheat and Summer Wheat. As demonstrated in Experiment 5.12 Best Model: Messis, while the label hierarchies we utilized have improved model performance, unnecessary distinctions between similar-looking crops often result in the model predominantly identifying the more common class, in this case, Winter Wheat. Additionally, training separate models for different seasons could be beneficial, given that for example Winter Wheat is typically harvested in late spring or early summer, whereas Summer Wheat is harvested from mid-summer to early fall (EOS Data Analytics, 2023). Such a seasonal model could improve classification accuracy by aligning the model's focus with the distinct growth periods and harvesting times of the crops.

Taking a step back, it would be valuable to test how a geospatial foundation model trained on European or even global data performs within the Messis framework. Despite Prithvi's training on a diverse range of landscapes in the United States (Jakubik et al., 2023), the somewhat different characteristics of European and Swiss environments may not be fully captured. This assessment would determine whether it's crucial for a geospatial model to be trained on global data, or if a diverse region like the United States can sufficiently capture the nuances of countries like Switzerland. Additionally, building crop classification models that do not only perform well in one specific region, requires more comprehensive ground truth data. An important recent development in this area is the EuroCrop dataset, a harmonized open crop dataset across the European Union, which offers a rich source of geo-referenced agricultural data that could significantly improve model performance in European contexts (Schneider et al., 2023).

Furthermore, given the modular nature of the Messis framework with its DVC pipeline to produce the dataset, extending the dataset to include additional cantons in Switzerland should be a straightforward process. This is because all cantons are required to report agricultural data and make it publicly available. Additionally, dealing with GeoTIFF overlaps can be managed by treating the GeoTIFF of each canton separately when slicing chips, avoiding the need to reproject and combine everything into one large image. The dataset expansion would provide more data points and offer a more comprehensive representation of the diverse agricultural regions within Switzerland. This could prove to be valuable for enhancing the performance of minority classes, as we have observed a clear tendency for better performance when more samples for a class are provided.

Another avenue for future work involves addressing a potential flaw in how hierarchical learning is currently approached in Messis, compared to the ms-ConvSTAR model. As pointed out in Section 6.2 RQ 2: How does Messis compare against ms-ConvSTAR?, unlike ms-ConvSTAR, which uses a series of convSTAR cells in each hierarchical layer to sequentially process spatio-temporal features through convolutional and recurrent operations, and thereby learns hierarchical features at each level, Messis relies on the Prithvi Vision Transformer (ViT) for feature extraction. Messis has no way of processing data sequentially between each hierarchy level and relies solely on convolutions for classification in the heads. This difference in architecture could affect the performance of Messis and its ability to fully utilize hierarchical information. Exploring this further seems especially relevant as we observed a significant degradation in precision and recall metrics for Tiers 2 and 3 in Messis, while its Tier 1 performance is significantly better than that of ms-ConvSTAR, as Experiment 5.12 Best Model: Messis shows. One potential solution is to modify the Prithvi Vision Transformer (which consists of 12 blocks) by branching off features at intermediate transformer blocks, e.g. blocks 4, 8, and 12 for a three-tier hierarchy. By doing this, we would capture information at various levels of abstraction. Each of these feature sets would then be processed through separate necks and classification heads tailored to each hierarchical level. This approach aims to better utilize hierarchical information, though it requires fine-tuning since the Vision Transformer is not initially designed to provide such intermediate embeddings.

We had additional experiments planned but did not conduct them due to different prioritization given by resource and time constraints. Details of these planned experiments can be found in the Appendix D.5 Further Experiment Ideas.

As we look towards the future, it's important to acknowledge the limitations and potential avenues for improvement that remain unaddressed in our work. The custom dataset fold split strategy needs validation across diverse datasets. Exploring a broader range of data augmentation techniques could yield better results. Further investigation into optimizer performance, particularly AdamW with varied learning rates, is necessary. Additionally, applying parameter-efficient fine-tuning methods like Low-Rank Adaptation (LoRA; Hu et al., 2021) to the model architecture warrants further exploration, especially when larger geospatial foundation models are published, that do not fit on consumer hardware.

# 8 Conclusion

With Messis, we successfully built a crop classification model by adapting the geospatial foundation model Prithvi, a pretrained multi-temporal vision transformer, on a slightly modified ZueriCrop dataset. This dataset is used for a crop classification task and includes more granular S2 satellite images from Europe, a continent not seen during the pretraining of Prithvi, and features smaller field sizes. We leveraged an explicit, hierarchical tree structure of the classes, which are highly skewed and long-tailed in their distribution. Additionally, we adapted ZueriCrop to a new label hierarchy based on seasonality. While this new hierarchy outperformed the previous one, the smaller number of classes in the new hierarchy does not allow us to conclusively determine whether it benefits the model. Furthermore, we demonstrated that a higher temporal resolution improves crop classification performance, even though we used only nine timesteps, compared to Turkoglu et al. (2021) who used eight times more observations. Therefore, future work should focus on exploring an even higher temporal resolution.

We engineered a strong foundation for conducting these experiments by establishing a DVC pipeline, setting up PyTorch Lightning, and tracking numerous metrics with Weights & Biases (W&B). Our logged W&B metrics have become a treasure trove of interesting insights, many of which we could simply not capture within the scope of this work. Additionally, we developed a web app that allows users to interactively try the model[16].

The successful adaptation of the geospatial foundation model highlights the versatility of such models in the agriculture and remote sensing domain. These geospatial models, in combination with readily available satellite imagery, can significantly aid farmers, agronomists, statisticians, and governments in developing more accurate machine-learning models with less data.

We proposed a new method to split multi-label data with counts into stratified folds. While it has achieved comparable performance to other approaches on our crop classification task, it lacks performance evaluations on more datasets and in other domains. Future work should focus on examining this approach on more datasets.

The advancements for foundation models in crop classification can significantly impact the agriculture industry and politics by enabling more data-driven agricultural policies and practices. Improved classification accuracy enables better agricultural practices, agricultural statistics, crop rotation, monitoring field activities in real-time, and subsidy validation, leading to increased efficiency in agriculture, which can influence policy decisions on subsidies, resource allocation, and environmental regulations. Additionally, the adoption of these technologies may shift the competitive landscape, favoring companies that invest in and leverage geospatial foundation models and remote sensing.

This work underscores the transformative potential of integrating geospatial foundation models in agriculture, paving the way for more efficient and accurate practices. Our findings not only enhance the understanding of geospatial foundation models and their usefulness, but also set a precedent for future innovation at the intersection of foundation models, agriculture, and remote sensing.

[16] https://huggingface.co/spaces/crop-classification/messis-demo

## References


Agence de services et de paiement Francaise. (2024, July). Système de suivi des surfaces agricoles en temps réel. Retrieved July 29, 2024, from https://www.asp-public.fr/missions-et-expertise/missions/pac-2023-2027/systeme-de-suivi-des-surfaces-agricoles-en-temps-reel

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021, November). ViViT: A Video Vision Transformer [arXiv:2103.15691 [cs]]. https://doi.org/10.48550/arXiv.2103.15691

Astola, H., Häme, T., Sirro, L., Molinier, M., & Kilpi, J. (2019). Comparison of Sentinel-2 and Landsat 8 imagery for forest variable prediction in boreal region. *Remote Sensing of Environment*, *223*, 257–273. https://doi.org/10.1016/j.rse.2019.01.019

BLW, B. f. L. (2024, July). Einzelkulturbeiträge. Retrieved July 31, 2024, from https://www.blw.admin.ch/blw/de/home/instrumente/direktzahlungen/einzelkulturbeitraege.html

Bundesamt für Landestopografie. (2022, October). Landwirtschaftliche Kulturflächen LNF Reference. Retrieved July 29, 2024, from https://geobasisdaten.ch/detail/818418/

Bundesamt für Landestopografie swisstopo. (2024, August). swissTLM3D. Retrieved August 10, 2024, from https://www.swisstopo.admin.ch/de/landschaftsmodell-swisstlm3d

Bundesamt für Landwirtschaft BLW. (2022, October). Landwirtschaftliche Kulturflächen. Retrieved July 29, 2024, from https://www.blw.admin.ch/blw/de/home/politik/datenmanagement/geografisches-informationssystem-gis/landwirtschaftliche-kulturflaechen.html

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., & Le, Q. V. (2023, May). Symbolic Discovery of Optimization Algorithms [arXiv:2302.06675 [cs]]. https://doi.org/10.48550/arXiv.2302.06675

Cherlinka, V. (2022, December). Types Of Crops: How To Classify And Manage Different Plants. Retrieved August 11, 2024, from https://eos.com/blog/types-of-crops/

Dave, P. R., & Pandya, H. A. (2020). Satellite Image Classification with Data Augmentation and Convolutional Neural Network. In T. Sengodan, M. Murugappan, & S. Misra (Eds.), *Advances in Electrical and Computer Technologies* (pp. 83–92). Springer Singapore.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [arXiv:1810.04805 [cs]]. https://doi.org/10.48550/arXiv.1810.04805

Dong, Q., Gong, S., & Zhu, X. (2018, April). Imbalanced Deep Learning by Minority Class Incremental Rectification [arXiv:1804.10851 [cs]]. https://doi.org/10.48550/arXiv.1804.10851

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021, June). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [arXiv:2010.11929 [cs]]. https://doi.org/10.48550/arXiv.2010.11929

EOS Data Analytics. (2023, May). Growing Wheat: Optimal Conditions, Timing, & Techniques. Retrieved August 5, 2024, from https://eos.com/blog/growing-wheat/

ESA. (n.d. a). S2 Mission. Retrieved July 10, 2024, from https://sentiwiki.copernicus.eu/web/s2-mission

ESA. (n.d. b). S2 Products. Retrieved July 29, 2024, from https://sentiwiki.copernicus.eu/web/s2-products

Eskesen, S., & Nyborg, L. (2021). Agricultural Control Using Sentinel.

European Commission. (2024, February). Simplification proposals to reduce administrative burdens *. Retrieved August 11, 2024, from https://ec.europa.eu/commission/presscorner/detail/ne/ip_24_1002

Forkuor, G., Dimobe, K., Serme, I., & Tondoh, J. E. (2018). Landsat-8 vs. Sentinel-2: Examining the added value of sentinel-2's red-edge bands to land-use and land-cover mapping in Burkina Faso [Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15481603.2017.1370169]. *GIScience & Remote Sensing*, *55*(3), 331–354. https://doi.org/10.1080/15481603.2017.1370169

Fraccaro, P., Gomes, C., Jakubik, J., Chu, L., Gabby, N., Bangalore, R., Lambhate, D., Das, K., Oliveira Borges, D., Kimura, D., Simumba, N., Szwarcman, D., Muszynski, M., Weldemariam, K., Edwards, B., Schmude, J., Hamann, H., Zadrozny, B., Ganti, R., ... Granger, E. (2023, August). HLS Foundation [original-date: 2023-07-11T15:05:32Z]. https://doi.org/https://huggingface.co/ibm-nasa-geospatial/Prithvi-100M

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks [ISSN: 1938-7228]. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256. Retrieved August 7, 2024, from https://proceedings.mlr.press/v9/glorot10a.html

Gregor Perich. (2024a, May). Weekly meeting vom 2. Mai 2024.

Gregor Perich. (2024b, July). Fragen zum Use-Case von Crop-Classification.

Gross, G., Helder, D., Begeman, C., Leigh, L., Kaewmanee, M., & Shah, R. (2022). Initial Cross-Calibration of Landsat 8 and Landsat 9 Using the Simultaneous Underfly Event [Number: 10 Publisher: Multidisciplinary Digital Publishing Institute]. *Remote Sensing*, *14*(10), 2418. https://doi.org/10.3390/rs14102418

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2021, December). Masked Autoencoders Are Scalable Vision Learners [arXiv:2111.06377 [cs]]. https://doi.org/10.48550/arXiv.2111.06377

He, K., Zhang, X., Ren, S., & Sun, J. (2015, December). Deep Residual Learning for Image Recognition [arXiv:1512.03385 [cs]]. Retrieved July 21, 2024, from http://arxiv.org/abs/1512.03385

He, X., Li, C., Zhang, P., Yang, J., & Wang, X. E. (2023). Parameter-Efficient Model Adaptation for Vision Transformers [Section: AAAI Technical Track on Computer Vision I]. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(1), 817–825. https://doi.org/10.1609/aaai.v37i1.25160

Henits, L., Szerletics, Á., Szokol, D., Szlovák, G., Gojdár, E., & Zlinszky, A. (2022). Sentinel-2 Enables Nationwide Monitoring of Single Area Payment Scheme and Greening Agricultural Subsidies in Hungary [Number: 16 Publisher: Multidisciplinary Digital Publishing Institute]. *Remote Sensing*, *14*(16), 3917. https://doi.org/10.3390/rs14163917

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models [_eprint: 2106.09685]. https://arxiv.org/abs/2106.09685

Jakubik, J., Roy, S., Phillips, C. E., Fraccaro, P., Godwin, D., Zadrozny, B., Szwarcman, D., Gomes, C., Nyirjesy, G., Edwards, B., Kimura, D., Simumba, N., Chu, L., Mukkavilli, S. K., Lambhate, D., Das, K., Bangalore, R., Oliveira, D., Muszynski, M., ... Ramachandran, R. (2023, November). Foundation Models for Generalist Geospatial Artificial Intelligence [arXiv:2310.18660 [cs]]. https://doi.org/10.48550/arXiv.2310.18660

Kingma, D. P., & Ba, J. (2017, January). Adam: A Method for Stochastic Optimization [arXiv:1412.6980 [cs]]. https://doi.org/10.48550/arXiv.1412.6980

Kolides, A., Nawaz, A., Rathor, A., Beeman, D., Hashmi, M., Fatima, S., Berdik, D., Al-Ayyoub, M., & Jararweh, Y. (2023). Artificial intelligence foundation and pre-trained models: Fundamentals, applications, opportunities, and social impacts. *Simulation Modelling Practice and Theory*, *126*, 102754. https://doi.org/https://doi.org/10.1016/j.simpat.2023.102754

Kumar, A., Shen, R., Bubeck, S., & Gunasekar, S. (2023, October). How to Fine-Tune Vision Models with SGD [arXiv:2211.09359 [cs]]. Retrieved June 11, 2024, from http://arxiv.org/abs/2211.09359

Landsat Science Outreach Team. (2021, November). Landsat 9 | Landsat Science. Retrieved July 10, 2024, from https://landsat.gsfc.nasa.gov/satellites/landsat-9/

Llugsi, R., Yacoubi, S. E., Fontaine, A., & Lupera, P. (2021). Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito. *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, 1–6. https://doi.org/10.1109/ETCM53643.2021.9590681

Loshchilov, I., & Hutter, F. (2019, January). Decoupled Weight Decay Regularization [arXiv:1711.05101 [cs, math]]. https://doi.org/10.48550/arXiv.1711.05101

McHugh, M. L. (2012). Interrater reliability: The kappa statistic [Publisher: Croatian Society of Medical Biochemistry and Laboratory Medicine]. *Biochemia Medica*, *22*(3), 276–282. https://doi.org/10.11613/BM.2012.031

NASA. (2022, October). Harmonized Landsat and Sentinel-2 (HLS) [Publisher: Earth Science Data Systems, NASA]. Retrieved July 29, 2024, from https://www.earthdata.nasa.gov/esds/harmonized-landsat-sentinel-2

Oubara, A., Wu, F., Amamra, A., & Yang, G. (2022). Survey on Remote Sensing Data Augmentation: Advances, Challenges, and Future Perspectives. In M. R. Senouci, S. Y. Boulahia, & M. A. Benatia (Eds.), *Advances in Computing Systems and Applications* (pp. 95–104). Springer International Publishing. https://doi.org/10.1007/978-3-031-12097-8_9

Patricia Rutz. (2023). Wann wird Getreide geerntet? *Goldküste24*. Retrieved August 12, 2024, from https://goldkueste24.ch/articles/196972-wann-wird-getreide-geerntet

Paul, S., & Kumar, D. N. (2019). COMPARISON OF LANDSAT-8 AND SENTINEL-2 DATA FOR CLASSIFICATION OF RABI CROPS OVER KARNATAKA, INDIA. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-3/W6*, 579–584. https://doi.org/10.5194/isprs-archives-XLII-3-W6-579-2019

Pawara, P., Okafor, E., Schomaker, L., & Wiering, M. (2017). Data Augmentation for Plant Classification. In J. Blanc-Talon, R. Penne, W. Philips, D. Popescu, & P. Scheunders (Eds.), *Advanced Concepts for Intelligent Vision Systems* (pp. 615–626). Springer International Publishing.

Pelletier, C., Webb, G. I., & Petitjean, F. (2019, January). Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series [arXiv:1811.10166 [cs]]. https://doi.org/10.48550/arXiv.1811.10166

Perich, G. (2023). *Satellite-based estimation of crop nitrogen and yield in Switzerland* [Doctoral Thesis]. ETH Zurich [Accepted: 2023-06-15T06:20:21Z]. https://doi.org/10.3929/ethz-b-000616562

Ruder, S. (2017, June). An overview of gradient descent optimization algorithms [arXiv:1609.04747 [cs]]. Retrieved July 18, 2024, from http://arxiv.org/abs/1609.04747

Rußwurm, M., & Körner, M. (2018). Multi-Temporal Land Cover Classification with Sequential Recurrent Encoders [Number: 4 Publisher: Multidisciplinary Digital Publishing Institute]. *ISPRS International Journal of Geo-Information*, *7*(4), 129. https://doi.org/10.3390/ijgi7040129

Rußwurm, M., Pelletier, C., Zollner, M., Lefèvre, S., & Körner, M. (2020, May). BreizhCrops: A Time Series Dataset for Crop Type Mapping [arXiv:1905.11893 [cs, stat]]. https://doi.org/10.48550/arXiv.1905.11893

Schneider, M., Schelte, T., Schmitz, F., & Körner, M. (2023). EuroCrops: The Largest Harmonized Open Crop Dataset Across the European Union [Publisher: Nature Publishing Group]. *Scientific Data*, *10*(1), 612. https://doi.org/10.1038/s41597-023-02517-0

Schweizerische Eidgenossenschaft. (2024, May). Direktzahlungsverordnung. Retrieved July 31, 2024, from https://www.fedlex.admin.ch/eli/cc/2013/765/de

Sechidis, K., Tsoumakas, G., & Vlahavas, I. (2011). On the Stratification of Multi-label Data [Series Title: Lecture Notes in Computer Science]. In D. Gunopulos, T. Hofmann, D. Malerba, & M. Vazirgiannis (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 145–158, Vol. 6913). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23808-6_10

Sen4CAP. (2024, July). Sentinel for common agricultural policy. Retrieved July 30, 2024, from http://www.esa-sen4cap.org/

Sinergise. (2023, June). Sentinel-2 Bands. Retrieved July 29, 2024, from https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved July 11, 2024, from http://jmlr.org/papers/v15/srivastava14a.html

Statistics Poland. (2024, July). Results of the application of satellite remote sensing methods for the development of a preliminary estimate of the main agricultural and horticultural crops. Retrieved July 30, 2024, from https://stat.gov.pl/en/experimental-statistics/agriculture/results-of-the-application-of-satellite-remote-sensing-methods-for-the-development-of-a-preliminary-estimate-of-the-main-agricultural-and-horticultural-crops,1,1.html

Stefano Ermon. (2024). Peering into the Future with Geospatial Foundation Models. Retrieved August 12, 2024, from https://www.atlasai.co/blog/foundation-models-for-geospatial-data

Swiss Parliament. (2022, May). 19.3988 | Digitalisierung im Agrarsektor. Rolle des Bundes | Geschäft | Das Schweizer Parlament. Retrieved July 30, 2024, from https://www.parlament.ch/de/ratsbetrieb/suche-curia-vista/geschaeft?AffairId=20193988

Toews, R. (2023, March). Transformers Revolutionized AI. What Will Replace Them? [Section: AI]. Retrieved June 29, 2024, from https://www.forbes.com/sites/robtoews/2023/09/03/transformers-revolutionized-ai-what-will-replace-them/

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015, June). Efficient Object Localization Using Convolutional Networks [arXiv:1411.4280 [cs]]. Retrieved July 13, 2024, from http://arxiv.org/abs/1411.4280

Turkoglu, M. O., D'Aronco, S., Perich, G., Liebisch, F., Streit, C., Schindler, K., & Wegner, J. D. (2021). Crop mapping from image time series: Deep learning with multi-scale label hierarchies. *Remote Sensing of Environment*, *264*, 112603. https://doi.org/10.1016/j.rse.2021.112603

Turkoglu, M. O., D'Aronco, S., Wegner, J. D., & Schindler, K. (2019, November). Gating Revisited: Deep Multi-layer RNNs That Can Be Trained [arXiv:1911.11033 [cs] version: 1]. https://doi.org/10.48550/arXiv.1911.11033

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023, August). Attention Is All You Need [arXiv:1706.03762 [cs]]. https://doi.org/10.48550/arXiv.1706.03762

Wang, P. (2024, July). Lucidrains/lion-pytorch [original-date: 2023-02-15T04:24:19Z]. Retrieved July 18, 2024, from https://github.com/lucidrains/lion-pytorch

Xin, Y., Luo, S., Zhou, H., Du, J., Liu, X., Fan, Y., Li, Q., & Du, Y. (2024). Parameter-Efficient Fine-Tuning for Pre-Trained Vision Models: A Survey [_eprint: 2402.02242]. https://arxiv.org/abs/2402.02242

Xingrui Yu, C. L., Xiaomin Wu, & Ren, P. (2017). Deep learning in remote sensing scene classification: A data augmentation enhanced convolutional neural network framework

[Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15481603.2017.1323377]. *GIScience & Remote Sensing*, *54*(5), 741–758. https://doi.org/10.1080/15481603.2017.1323377

Zhou, P., Xie, X., Lin, Z., & Yan, S. (2024). Towards Understanding Convergence and Generalization of AdamW. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–8. https://doi.org/10.1109/TPAMI.2024.3382294

# Appendices

# A Dataset

This appendix covers the definition of the different datasets used in our work and their label hierarchies in full detail as well as more detailed evaluations of the stratified fold split strategy.

## A.1 ZueriCrop 2.0 Dataset with ZueriCrop Label Hierarchy

The ZueriCrop 2.0 dataset with a label hierarchy equal to the one used in the original ZueriCrop dataset.

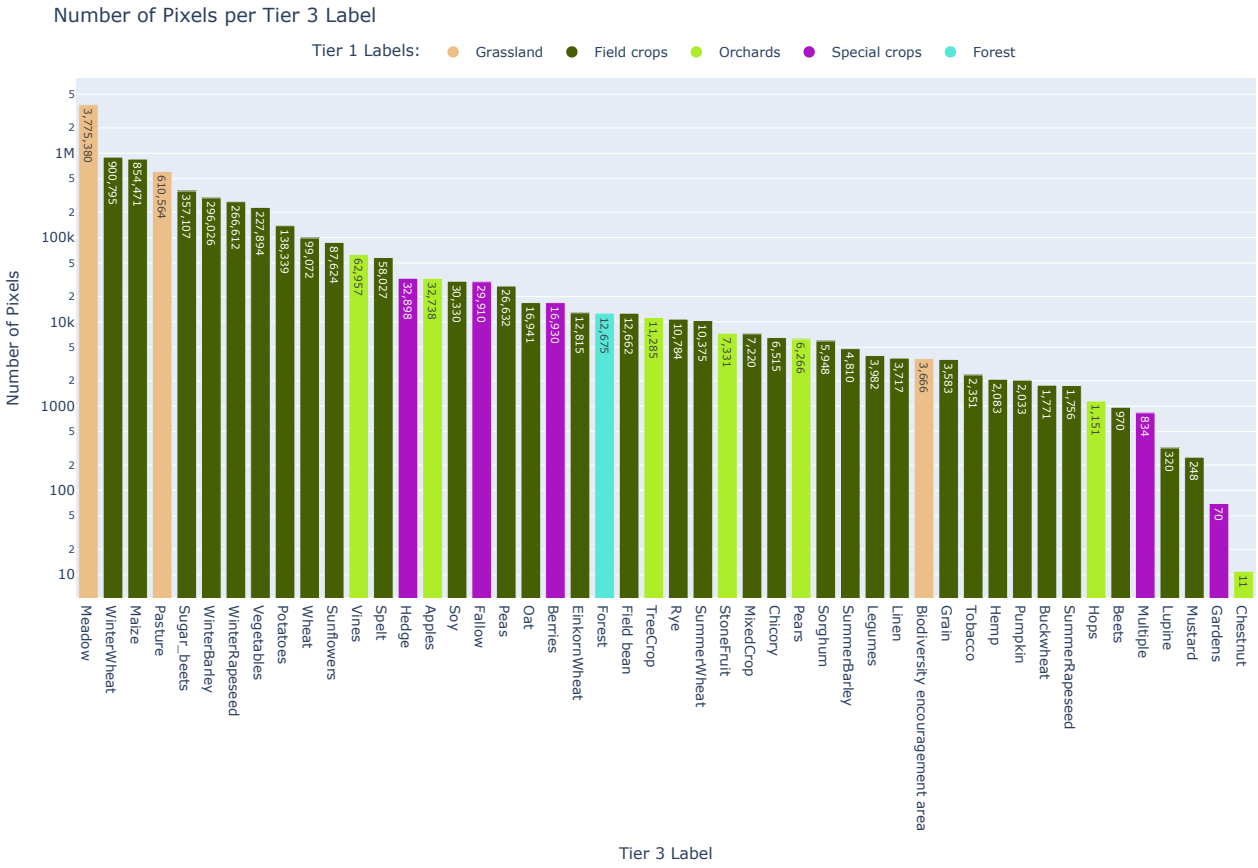



**Figure A.1:** ZueriCrop 2.0 Dataset class distribution in Tier 3 with ZueriCrop Hierarchy. Logarithmic scale.

**Table A.1:** The ZueriCrop 2.0 Label Hierarchy, including its three tiers and the associated LNF Codes for each of the 48 classes.

| Tier 1 | Tier 2 | Tier 3 | LNF Codes |
|---|---|---|---|
| Field crops | BroadLeafRowCrop | Beets | [523] |
| Field crops | BroadLeafRowCrop | Field bean | [536] |
| Field crops | BroadLeafRowCrop | Hemp | [535] |
| Field crops | BroadLeafRowCrop | Legumes | [631] |
| Field crops | BroadLeafRowCrop | Linen | [534, 544] |

Continued on next page

**Table A.1 – continued from previous page**

| Tier 1 | Tier 2 | Tier 3 | LNF Codes |
|---|---|---|---|
| Field crops | BroadLeafRowCrop | Lupine | [538] |
| Field crops | BroadLeafRowCrop | Mustard | [573] |
| Field crops | BroadLeafRowCrop | Peas | [537] |
| Field crops | BroadLeafRowCrop | Potatoes | [524, 525] |
| Field crops | BroadLeafRowCrop | Soy | [528] |
| Field crops | BroadLeafRowCrop | Sugar_beets | [522] |
| Field crops | BroadLeafRowCrop | SummerRapeseed | [526] |
| Field crops | BroadLeafRowCrop | Sunflowers | [531, 592] |
| Field crops | BroadLeafRowCrop | Tobacco | [541] |
| Field crops | BroadLeafRowCrop | WinterRapeseed | [527, 591] |
| Field crops | CropMix | MixedCrop | [569] |
| Field crops | LargeGrainCereal | Maize | [508, 519, 521] |
| Field crops | LargeGrainCereal | Sorghum | [542, 549] |
| Field crops | SmallGrainCereal | Buckwheat | [548] |
| Field crops | SmallGrainCereal | EinkornWheat | [511] |
| Field crops | SmallGrainCereal | Grain | [506, 515, 543] |
| Field crops | SmallGrainCereal | Oat | [504] |
| Field crops | SmallGrainCereal | Rye | [514] |
| Field crops | SmallGrainCereal | Spelt | [516] |
| Field crops | SmallGrainCereal | SummerBarley | [501] |
| Field crops | SmallGrainCereal | SummerWheat | [512] |
| Field crops | SmallGrainCereal | Wheat | [505, 507] |
| Field crops | SmallGrainCereal | WinterBarley | [502] |
| Field crops | SmallGrainCereal | WinterWheat | [513] |
| Field crops | VegetableCrop | Chicory | [547] |
| Field crops | VegetableCrop | Pumpkin | [539] |
| Field crops | VegetableCrop | Vegetables | [545, 546] |
| Forest | Forest | Forest | [901] |
| Grassland | BiodiversityArea | Biodiversity encouragement area | [555, 572, 908] |
| Grassland | Meadow | Meadow | [601, 602, 611, 612, 613, 621, 622, 634] |
| Grassland | Pasture | Pasture | [616, 617, 618, 625, 930] |
| Orchards | OrchardCrop | Apples | [702] |
| Orchards | OrchardCrop | Chestnut | [720] |
| Orchards | OrchardCrop | Hops | [708] |
| Orchards | OrchardCrop | Pears | [703] |
| Orchards | OrchardCrop | StoneFruit | [704] |
| Orchards | OrchardCrop | Vines | [701, 717, 722, 735] |
| Orchards | TreeCrop | TreeCrop | [712, 713] |
| Special crops | Berries | Berries | [551, 705] |
| Special crops | Fallow | Fallow | [556, 557] |
| Special crops | Gardens | Gardens | [909] |
| Special crops | Hedge | Hedge | [852, 857, 858] |
| Special crops | Multiple | Multiple | [902] |

## A.2 ZueriCrop 2.0 Dataset with Seasonality Label Hierarchy

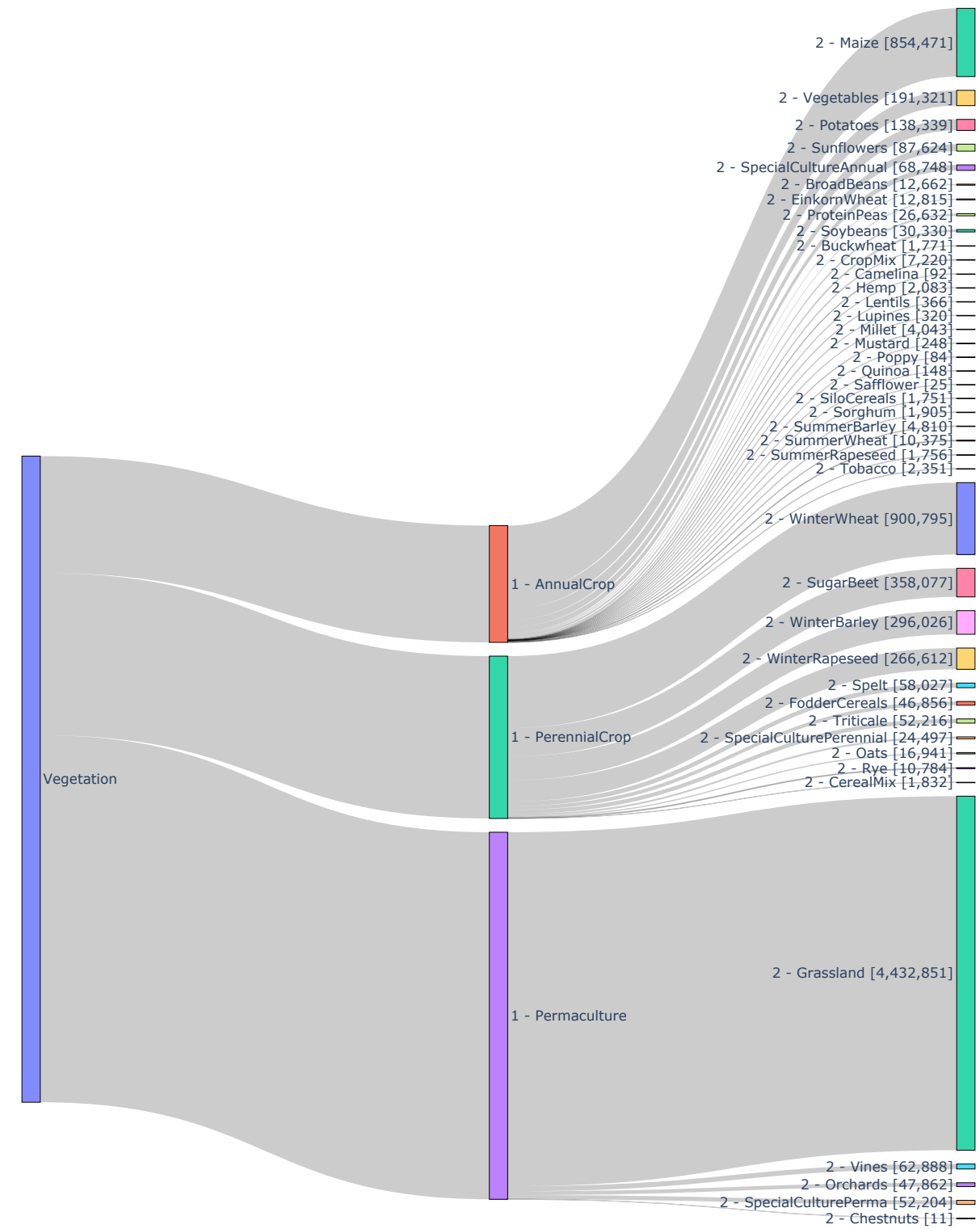



**Figure A.2:** Sankey diagram of the ZueriCrop 2.0 dataset in the newly proposed Seasonality Hierarchy.

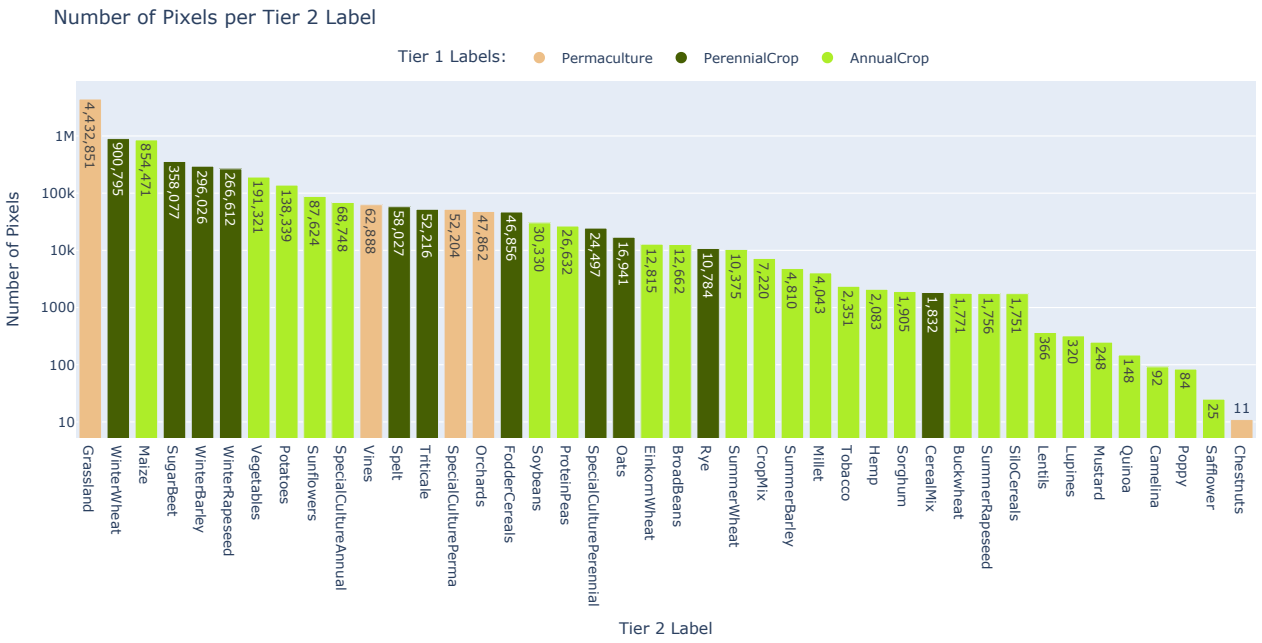



**Figure A.3:** ZueriCrop 2.0 Dataset class distribution in Tier 2 with Seasonality Hierarchy. Logarithmic scale.

**Table A.2:** The Seasonality Label Hierarchy, including its two tiers and the associated LNF Codes for each of the 42 classes.

| **Tier 1** | **Tier 2** | **LNF Codes** |
|---|---|---|
| AnnualCrop | BroadBeans | [536] |
| AnnualCrop | Buckwheat | [548] |
| AnnualCrop | Camelina | [544] |
| AnnualCrop | CropMix | [569, 570] |
| AnnualCrop | EinkornWheat | [511] |
| AnnualCrop | Hemp | [535, 575, 576, 577] |
| AnnualCrop | Lentils | [568] |
| AnnualCrop | Lupines | [538] |
| AnnualCrop | Maize | [508, 519, 521] |
| AnnualCrop | Millet | [542, 578, 579] |
| AnnualCrop | Mustard | [573] |
| AnnualCrop | Poppy | [566] |
| AnnualCrop | Potatoes | [524, 525] |
| AnnualCrop | ProteinPeas | [537] |
| AnnualCrop | Quinoa | [574] |
| AnnualCrop | Safflower | [567] |
| AnnualCrop | SiloCereals | [543] |
| AnnualCrop | Sorghum | [549, 580, 581] |
| AnnualCrop | Soybeans | [528] |
| AnnualCrop | SpecialCultureAnnual | [534, 539, 546, 547, 551, 552, 553, 554, 631, 709, 711, 951] |
| AnnualCrop | SummerBarley | [501] |
| AnnualCrop | SummerRapeseed | [526, 590] |
| AnnualCrop | SummerWheat | [512] |
| AnnualCrop | Sunflowers | [531, 592] |
| AnnualCrop | Tobacco | [541] |
| AnnualCrop | Vegetables | [545] |

Continued on next page

**Table A.2 – continued from previous page**

| Tier 1 | Tier 2 | LNF Codes |
|---|---|---|
| PerennialCrop | CerealMix | [506, 515] |
| PerennialCrop | FodderCereals | [507] |
| PerennialCrop | Oats | [504] |
| PerennialCrop | Rye | [514] |
| PerennialCrop | SpecialCulturePerennial | [705, 706, 707, 710] |
| PerennialCrop | Spelt | [516] |
| PerennialCrop | SugarBeet | [522, 523] |
| PerennialCrop | Triticale | [505] |
| PerennialCrop | WinterBarley | [502] |
| PerennialCrop | WinterRapeseed | [527, 591] |
| PerennialCrop | WinterWheat | [513] |
| Permaculture | Chestnuts | [720, 923] |
| Permaculture | Grassland | [555, 556, 557, 559, 572, 601, 602, 611, 612, 613, 616, 617, 618, 621, 622, 623, 625, 632, 634, 635, 650, 660, 693, 694, 697, 698, 750, 760, 797, 798, 908, 930, 933, 935, 950] |
| Permaculture | Orchards | [702, 703, 704, 718, 719, 722, 730, 731, 921, 922, 924, 927] |
| Permaculture | SpecialCulturePerma | [708, 712, 713, 714, 715, 721, 723, 724, 725, 852, 857, 858, 902, 909] |
| Permaculture | Vines | [701, 717, 735] |

## A.3 ZueriCrop 2.0 Dataset with Reduced Seasonality Label Hierarchy

The reduced seasonality-based hierarchy, where all classes with less than 10'000 ground truth pixels present in the dataset for the cantons of Zürich and Thurgau in 2019 were dropped.

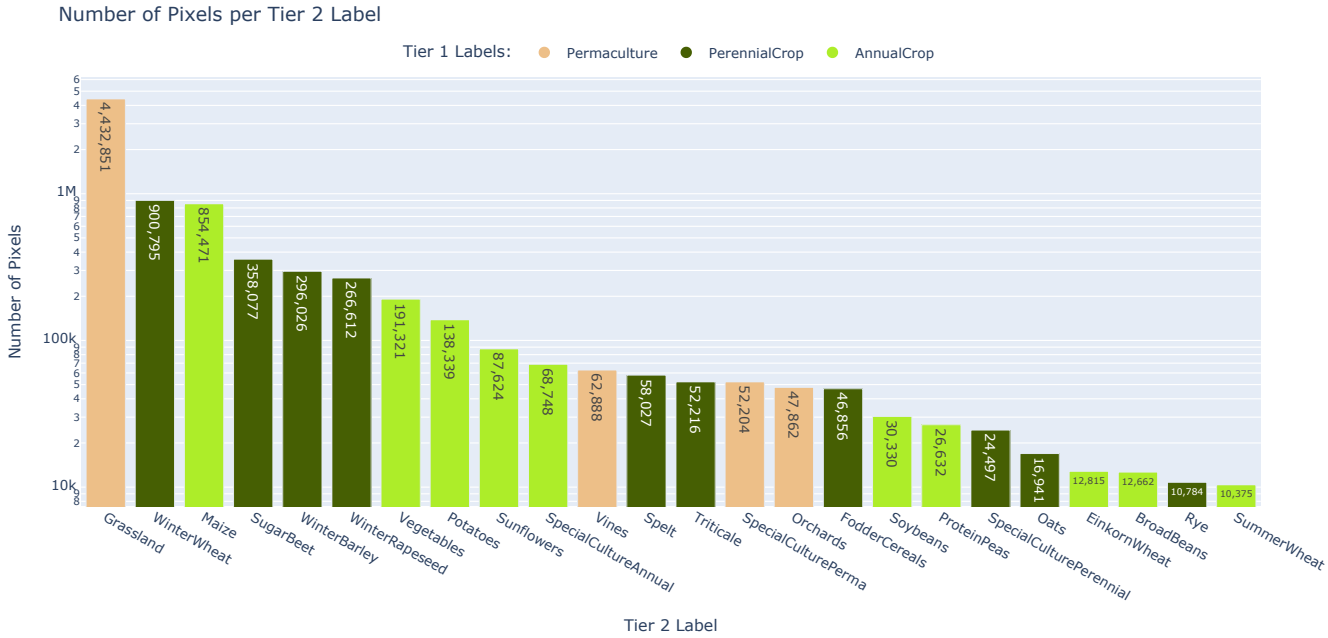



**Figure A.4:** ZueriCrop 2.0 Dataset class distribution in Tier 2 with Reduced Seasonality Hierarchy. Logarithmic scale.

**Table A.3:** The Reduced Seasonality Label Hierarchy, including its two tiers and the associated LNF Codes for each of the 24 classes.

| Tier 1 | Tier 2 | LNF Codes |
|---|---|---|
| AnnualCrop | BroadBeans | [536] |
| AnnualCrop | Buckwheat | [548] |
| AnnualCrop | Camelina | [544] |
| AnnualCrop | CropMix | [569, 570] |
| AnnualCrop | EinkornWheat | [511] |
| AnnualCrop | Hemp | [535, 575, 576, 577] |
| AnnualCrop | Lentils | [568] |
| AnnualCrop | Lupines | [538] |
| AnnualCrop | Maize | [508, 519, 521] |
| AnnualCrop | Millet | [542, 578, 579] |
| AnnualCrop | Mustard | [573] |
| AnnualCrop | Poppy | [566] |
| AnnualCrop | Potatoes | [524, 525] |
| AnnualCrop | ProteinPeas | [537] |
| AnnualCrop | Quinoa | [574] |
| AnnualCrop | Safflower | [567] |
| AnnualCrop | SiloCereals | [543] |
| AnnualCrop | Sorghum | [549, 580, 581] |
| AnnualCrop | Soybeans | [528] |
| AnnualCrop | SpecialCultureAnnual | [534, 539, 546, 547, 551, 552, 553, 554, 631, 709, 711, 951] |
| AnnualCrop | SummerBarley | [501] |
| AnnualCrop | SummerRapeseed | [526, 590] |

Continued on next page

**Table A.3 – continued from previous page**

| Tier 1 | Tier 2 | LNF Codes |
|---|---|---|
| AnnualCrop | SummerWheat | [512] |
| AnnualCrop | Sunflowers | [531, 592] |
| AnnualCrop | Tobacco | [541] |
| AnnualCrop | Vegetables | [545] |
| PerennialCrop | CerealMix | [506, 515] |
| PerennialCrop | FodderCereals | [507] |
| PerennialCrop | Oats | [504] |
| PerennialCrop | Rye | [514] |
| PerennialCrop | SpecialCulturePerennial | [705, 706, 707, 710] |
| PerennialCrop | Spelt | [516] |
| PerennialCrop | SugarBeet | [522, 523] |
| PerennialCrop | Triticale | [505] |
| PerennialCrop | WinterBarley | [502] |
| PerennialCrop | WinterRapeseed | [527, 591] |
| PerennialCrop | WinterWheat | [513] |
| Permaculture | Chestnuts | [720, 923] |
| Permaculture | Grassland | [555, 556, 557, 559, 572, 601, 602, 611, 612, 613, 616, 617, 618, 621, 622, 623, 625, 632, 634, 635, 650, 660, 693, 694, 697, 698, 750, 760, 797, 798, 908, 930, 933, 935, 950] |
| Permaculture | Orchards | [702, 703, 704, 718, 719, 722, 730, 731, 921, 922, 924, 927] |
| Permaculture | SpecialCulturePerma | [708, 712, 713, 714, 715, 721, 723, 724, 725, 852, 857, 858, 902, 909] |
| Permaculture | Vines | [701, 717, 735] |

## A.4 LNF Code Reference Table

The following reference table can be used to understand what each LNF Code listed in the hierarchies stands for (Bundesamt für Landestopografie, 2022).

**Table A.4:** Lookup code table for the LNF codes and their corresponding description in German.

| LNF Code | Utilization (DE) |
|---|---|
| 501 | Sommergerste |
| 502 | Wintergerste |
| 504 | Hafer |
| 505 | Triticale |
| 506 | Mischel Futtergetreide |
| 507 | Futterweizen gemäss Sortenliste swiss granum |
| 508 | Körnermais |
| 509 | Reis |
| 510 | Hartweizen |
| 511 | Emmer, Einkorn |
| 512 | Sommerweizen (ohne Futterweizen der Sortenliste swiss granum) |

Continued on next page

**Table A.4 – continued from previous page**

| LNF Code | Utilization (DE) |
|---|---|
| 513 | Winterweizen (ohne Futterweizen der Sortenliste swiss granum) |
| 514 | Roggen |
| 515 | Mischel Brotgetreide |
| 516 | Dinkel |
| 519 | Saatmais (Vertragsanbau) |
| 520 | Trockenreis |
| 521 | Silo- und Grünmais |
| 522 | Zuckerrüben |
| 523 | Futterrüben |
| 524 | Kartoffeln |
| 525 | Pflanzkartoffeln (Vertragsanbau) |
| 526 | Sommerraps zur Speiseölgewinnung |
| 527 | Winterraps zur Speiseölgewinnung |
| 528 | Soja |
| 529 | Nassreis |
| 531 | Sonnenblumen zur Speiseölgewinnung |
| 534 | Lein |
| 535 | Hanf |
| 536 | Bohnen und Wicken zur Körnergewinnung (z.B. Ackerbohnen) |
| 537 | Erbsen zur Körnergewinnung (z.B. Eiweisserbsen) |
| 538 | Lupinen |
| 539 | Ölkürbisse |
| 540 | Kichererbsen |
| 541 | Tabak |
| 542 | Hirse |
| 543 | Getreide siliert |
| 544 | Leindotter |
| 545 | Einjährige Freilandgemüse, ohne Konservengemüse |
| 546 | Freiland-Konservengemüse |
| 547 | Wurzeln der Treibzichorie |
| 548 | Buchweizen |
| 549 | Sorghum |
| 550 | Übrige Ackerfläche |
| 551 | Einjährige Beeren (z.B. Erdbeeren) |
| 552 | Einjährige nachwachsende Rohstoffe (Kenaf, usw.) |
| 553 | Einjährige Gewürz- und Medizinalpflanzen |
| 554 | Einjährige gärtnerische Freilandkulturen (Blumen, Rollrasen usw.) |
| 555 | Ackerschonstreifen |
| 556 | Buntbrache |
| 557 | Rotationsbrache |
| 559 | Saum auf Ackerflächen |
| 566 | Mohn |
| 567 | Saflor |
| 568 | Linsen |
| 569 | Mischungen von Bohnen, Wicken, Erbsen, Kichererbsen und Lupinen mit Getreide oder Leindotter, mindestens 30 % Anteil Leguminosen bei der Ernte (zur Körnergewinnung) |

Continued on next page

**Table A.4 – continued from previous page**

| LNF Code | Utilization (DE) |
|---|---|
| 570 | Mischungen von Linsen mit Getreide oder Leindotter, mindestens 30 % Anteil Linsen bei der Ernte (zur Körnergewinnung) |
| 572 | Nützlingsstreifen auf offener Ackerfläche |
| 573 | Senf |
| 574 | Quinoa |
| 575 | Hanf zur Nutzung der Samen |
| 576 | Hanf zur Fasernutzung |
| 577 | Anderer Hanf |
| 578 | Hirse zur Körnergewinnung |
| 579 | Hirse zur Nutzung ganze Pflanze |
| 580 | Sorghum zur Körnergewinnung |
| 581 | Sorghum zur Nutzung ganze Pflanze |
| 590 | Sommerraps als nachwachsender Rohstoff |
| 591 | Winterraps als nachwachsender Rohstoff |
| 592 | Sonnenblumen als nachwachsender Rohstoff |
| 594 | Offene Ackerfläche, beitragsberechtigt (regionsspezifische Biodiversitätsförderfläche) |
| 595 | Übrige offene Ackerfläche, nicht beitragsberechtigt (regionsspezifische Biodiversitätsförderfläche) |
| 597 | Übrige offene Ackerfläche, beitragsberechtigt |
| 598 | Übrige offene Ackerfläche, nicht beitragsberechtigt |
| 601 | Kunstwiesen (ohne Weiden) |
| 602 | Übrige Kunstwiese, beitragsberechtigt (z.B. Schweineweide, Geflügelweide) |
| 611 | Extensiv genutzte Wiesen (ohne Weiden) |
| 612 | Wenig intensiv genutzte Wiesen (ohne Weiden) |
| 613 | Übrige Dauerwiesen (ohne Weiden) |
| 616 | Weiden (Heimweiden, übrige Weiden ohne Sömmerungsweiden) |
| 617 | Extensiv genutzte Weiden |
| 618 | Waldweiden (ohne bewaldete Fläche) |
| 621 | Heuwiesen im Sömmerungsgebiet, Übrige Wiesen |
| 622 | Heuwiesen im Sömmerungsgebiet, Typ extensiv genutzte Wiese |
| 623 | Heuwiesen im Sömmerungsgebiet, Typ wenig intensiv genutzte Wiese |
| 625 | Waldweiden (ohne bewaldete Fläche) |
| 631 | Futterleguminosen für die Samenproduktion (Vertragsanbau) |
| 632 | Futtergräser für die Samenproduktion (Vertragsanbau) |
| 634 | Uferwiesen entlang von Fliessgewässern (ohne Weiden) |
| 635 | Uferwiesen (ohne Weiden) |
| 650 | Übrige Dauerwiesen, beitragsberechtigt aggregiert |
| 660 | Übrige Dauerweiden, beitragsberechtigt aggregiert |
| 693 | Regionsspezifische Biodiversitätsförderflächen (Weiden) |
| 694 | Regionsspezifische Biodiversitätsförderfläche (Grünflächen ohne Weiden) |
| 697 | Übrige Grünfläche (Dauergrünfläche), beitragsberechtigt |
| 698 | Übrige Grünfläche (Dauergrünflächen), nicht beitragsberechtigt |
| 701 | Reben |
| 702 | Obstanlagen (Äpfel) |

Continued on next page

**Table A.4 – continued from previous page**

| LNF Code | Utilization (DE) |
|---|---|
| 703 | Obstanlagen (Birnen) |
| 704 | Obstanlagen (Steinobst) |
| 705 | Mehrjährige Beeren |
| 706 | Mehrjährige Gewürz- und Medizinalpflanzen |
| 707 | Mehrjährige nachwachsende Rohstoffe (Chinaschilf, usw.) |
| 708 | Hopfen |
| 709 | Rhabarber |
| 710 | Spargel |
| 711 | Pilze (Freiland) |
| 712 | Christbäume |
| 713 | Baumschule von Forstpflanzen ausserhalb der Forstzone |
| 714 | Ziersträucher, Ziergehölze und Zierstauden |
| 715 | Übrige Baumschulen (Rosen, Früchte, usw.) |
| 717 | Rebflächen mit natürlicher Artenvielfalt |
| 718 | Trüffelanlagen |
| 719 | Maulbeerbaumanlagen (Fütterung Seidenraupen) |
| 720 | Gepflegte Selven (Edelkastanienbäume) |
| 721 | Mehrjährige gärtnerische Freilandkulturen (nicht im Gewächshaus) |
| 722 | Baumschulen von Reben |
| 723 | Baumschulen von Obst und Beeren |
| 724 | Übrige Baumschulen (Rosen, Zierstauden, usw.) |
| 725 | Permakultur |
| 730 | Obstanlagen aggregiert |
| 731 | Andere Obstanlagen (Kiwis, Holunder usw.) |
| 735 | Reben (regionsspezifische Biodiversitätsförderflächen) |
| 750 | Übrige Dauerkulturen, beitragsberechtigt, aggregiert |
| 760 | Dauerkulturen, nicht beitragsberechtigt, aggregiert |
| 797 | Übrige Flächen mit Dauerkulturen, beitragsberechtigt |
| 798 | Übrige Flächen mit Dauerkulturen, nicht beitragsberechtigt |
| 801 | Gemüsekulturen in Gewächshäusern mit festem Fundament |
| 802 | Übrige Spezialkulturen in Gewächshäusern mit festem Fundament |
| 803 | Gärtnerische Kulturen in Gewächshäusern mit festem Fundament |
| 804 | Beerenkulturen in Gewächshäusern mit festem Fundament |
| 806 | Gemüsekulturen in geschütztem Anbau ohne festes Fundament |
| 807 | Übrige Spezialkulturen in geschütztem Anbau ohne festes Fundament |
| 808 | Gärtnerische Kulturen in geschütztem Anbau ohne festes Fundament |
| 810 | Pilze in geschütztem Anbau mit festem Fundament |
| 811 | Gemüsekulturen in geschütztem Anbau ohne festes Fundament; im gewachsenen Boden |
| 812 | Gemüsekulturen in geschütztem Anbau ohne festes Fundament; auf Pflanztischen oder -gestellen |
| 813 | Beerenkulturen in geschütztem Anbau ohne festes Fundament; im gewachsenen Boden |
| 814 | Beerenkulturen in geschütztem Anbau ohne festes Fundament; auf Pflanztischen oder -gestellen |
| 830 | Kulturen in ganzjährig geschütztem Anbau, beitragsberechtigt aggregiert |

Continued on next page

**Table A.4 – continued from previous page**

| LNF Code | Utilization (DE) |
|---|---|
| 840 | Kulturen in ganzjährig geschütztem Anbau, nicht beitragsberechtigt aggregiert |
| 847 | Übrige Kulturen in geschütztem Anbau ohne festes Fundament, beitragsberechtigt |
| 848 | Übrige Kulturen in geschütztem Anbau mit festem Fundament |
| 849 | Übrige Kulturen in geschütztem Anbau ohne festes Fundament, nicht beitragsberechtigt |
| 851 | Streueflächen in der LN |
| 852 | Hecken-, Feld- und Ufergehölze (mit Krautsaum) |
| 857 | Hecken-, Feld- und Ufergehölze (mit Pufferstreifen) |
| 858 | Hecken-, Feld- und Ufergehölze (mit Pufferstreifen) (regionsspezifische Biodiversitätsförderfläche) |
| 897 | Übrige Flächen innerhalb der LN, beitragsberechtigt |
| 898 | Übrige Flächen innerhalb der LN, nicht beitragsberechtigt |
| 901 | Wald |
| 902 | Übrige unproduktive Flächen (z.B. gemulchte Flächen, stark verunkrautete Flächen, Hecken ohne Pufferstreifen) |
| 903 | Flächen ohne landwirtschaftliche Hauptzweckbestimmung (erschlossenes Bauland, Spiel-, Reit-, Camping-, Golf-, Flug- und Militärplätze oder ausgemarchte Bereiche von Eisenbahnen, öffentlichen Strassen und Gewässern) |
| 904 | Wassergräben, Tümpel, Teiche |
| 905 | Ruderalflächen, Steinhaufen und -wälle |
| 906 | Trockenmauern |
| 907 | Unbefestigte, natürliche Wege |
| 908 | Regionsspezifische Biodiversitätsförderflächen |
| 909 | Hausgärten |
| 911 | Landwirtschaftliche Produktion in Gebäuden (z. B. Champignon, Brüsseler) |
| 921 | Hochstamm-Feldobstbäume (Punkte oder Flächen) |
| 922 | Nussbäume (Punkte oder Flächen) |
| 923 | Edelkastanienbäume |
| 924 | Einheimische standortgerechte Einzelbäume und Alleen (Punkte oder Flächen) |
| 926 | Andere Bäume |
| 927 | Andere Bäume (regionsspezifische Biodiversitätsförderfläche) |
| 928 | Andere Elemente (regionsspezifische Biodiversitätsförderfläche) |
| 930 | Sömmerungsweiden |
| 933 | Gemeinschaftsweiden |
| 935 | Heuwiesen mit Zufütterung während der Sömmerung |
| 936 | Streueflächen im Sömmerungsgebiet |
| 950 | Ackerschonstreifen |
| 951 | Getreide in weiter Reihen |
| 998 | Übrige Flächen ausserhalb der LN und SF |

## A.5 Penalty for Stratified Split Strategy

| Formula | Mean kullback-leibler score |
|---|---|
| $\alpha = 1$ | 0.345 |
| $\alpha = \frac{1}{10}$ | 0.369 |
| $\alpha = \frac{1}{\log(n_{\text{chips}} - i + 1)}$ | 0.372 |
| $\alpha = \frac{1}{10 \log(n_{\text{chips}} - i + 1)}$ | <u>0.315</u> |
| $\alpha = \frac{1}{\sqrt{n_{\text{chips}} - i + 1}}$ | 0.359 |
| $\alpha = \frac{1}{3\sqrt{n_{\text{chips}} - i + 1}}$ | 0.370 |
| $\alpha = \frac{1}{10\sqrt{n_{\text{chips}} - i + 1}}$ | 0.339 |
| $\alpha = \frac{1}{30\sqrt{n_{\text{chips}} - i + 1}}$ | **0.152** |

**Table A.5:** Different formulas to calculate the $\alpha$ in the penalty term for our data split strategy described in Section 4.6.3 Stratified Strategy alongside with the final mean kullback-leibler score. The best score is highlighted in bold and the second best is underlined.

## A.6 Crop distributions of the 6-fold split

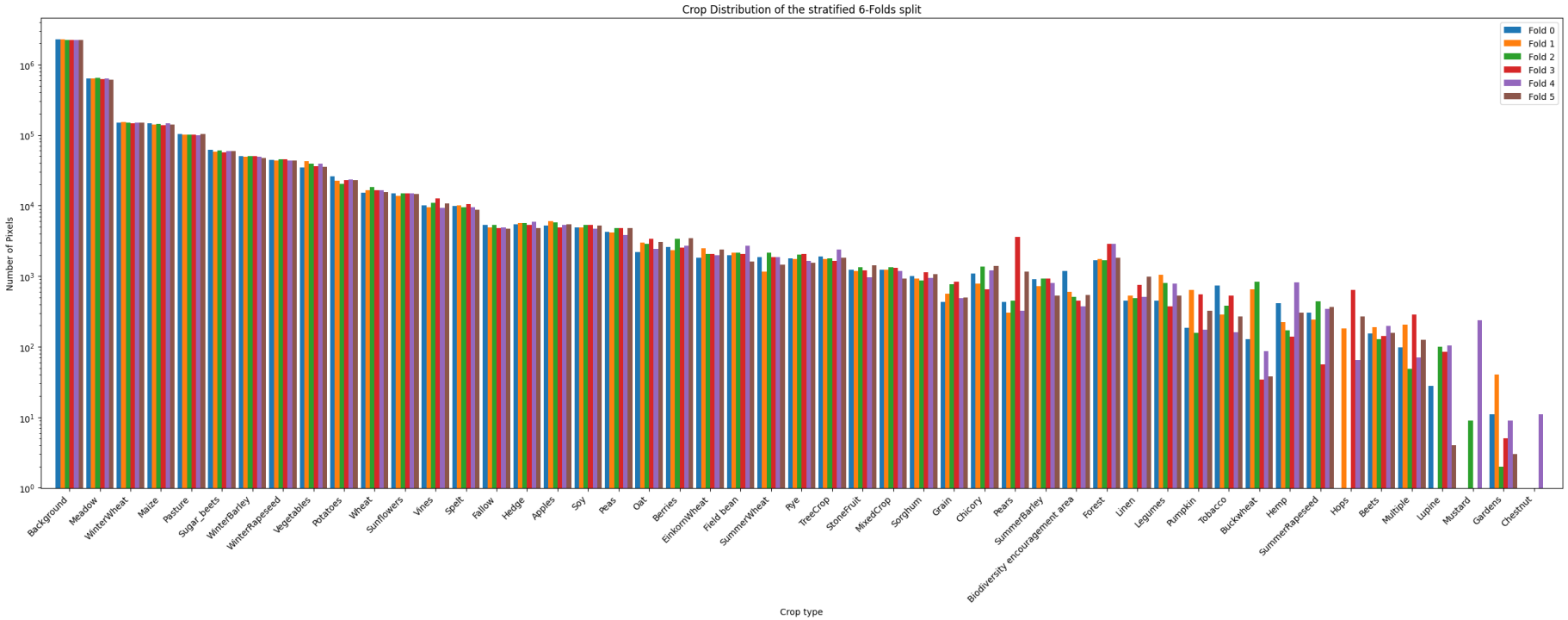
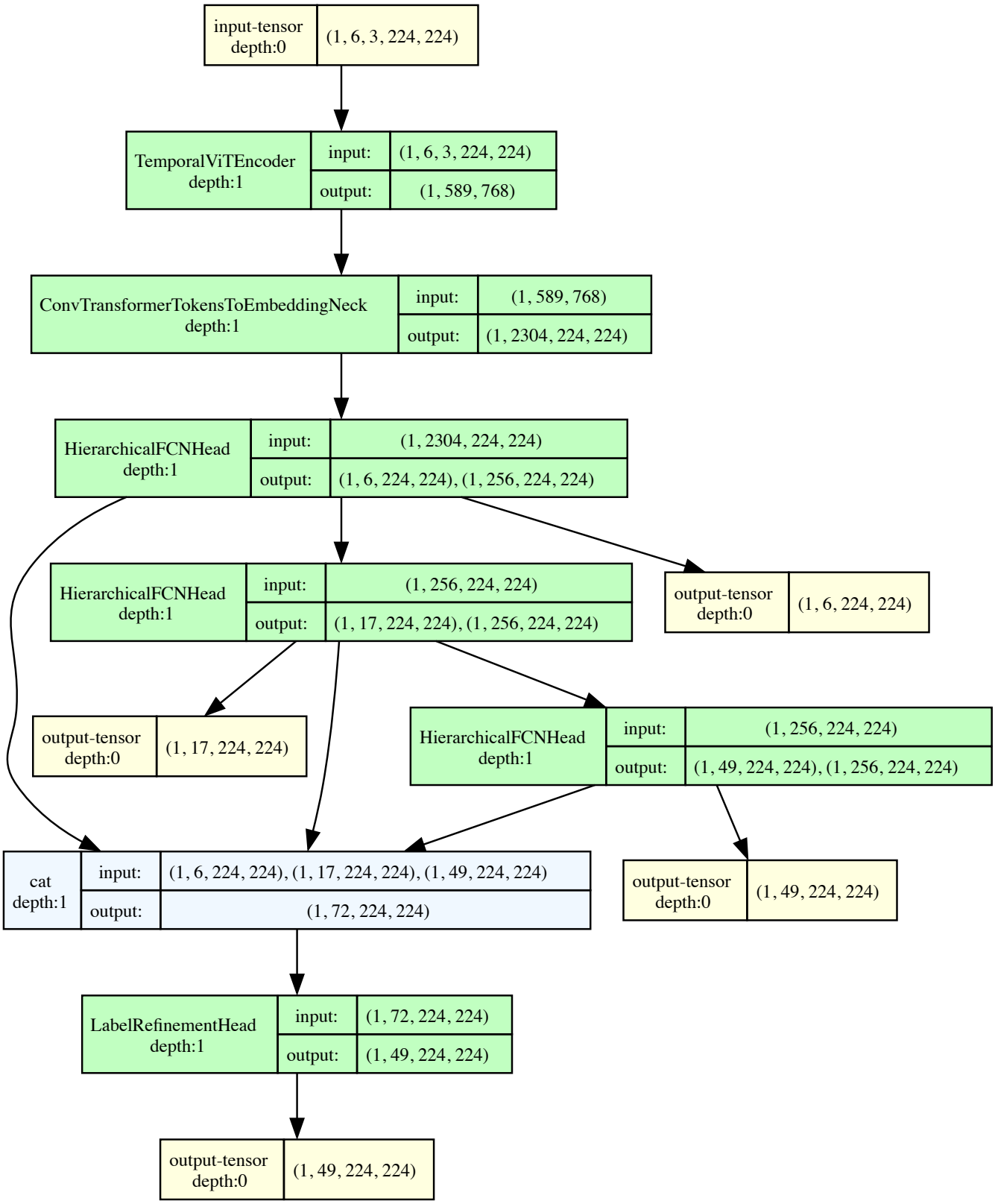



**Figure A.5:** Distribution of the (log) pixel counts for each crop type over 6 folds generated by our stratified dataset split approach as described in Section 4.6.3 Stratified Strategy. Note: 4 crops do not appear in all folds. Chestnut appears only once while lupine, hops, and mustard appear 5, 4, and 2 times respectively in the 6 folds.

# B Messis Architecture



# C Additional Results

This appendix contains additional results from our experiments that were not included in the main report.

## C.1 Alternative Label Hierarchy Results

This appendix presents the confusion matrix for Messis when trained using the seasonality-based label hierarchy, as in Experiment 5.7 Alternative Label Hierarchy.

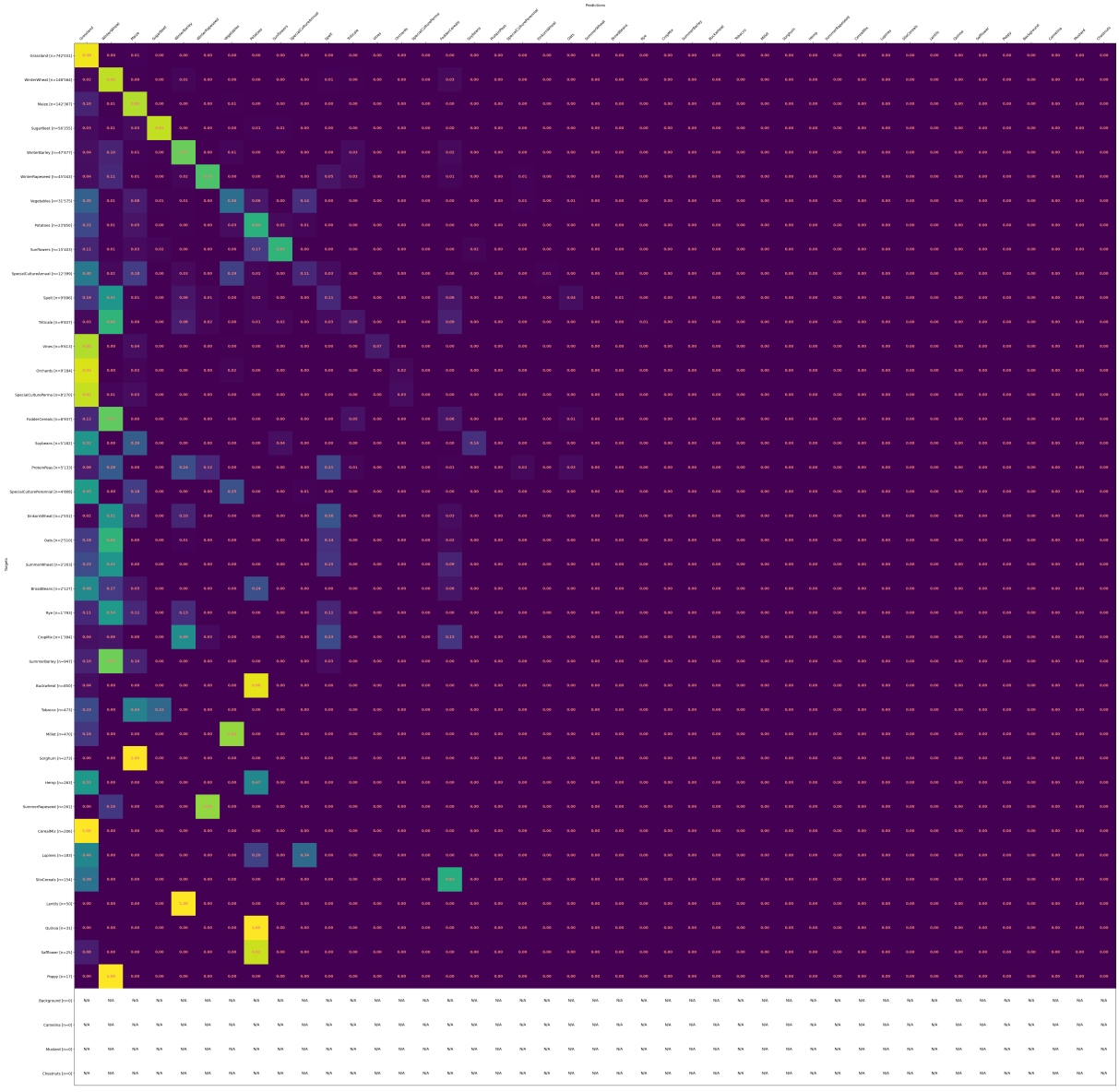


**Figure C.1:** Confusion matrix of Tier 2 for model trained on the seasonality hierarchy, calculated field majority with fold 0 as validation fold.

## C.2 Best Model Results

This appendix lists detailed performance metrics for our best model, calculated on the test fold.

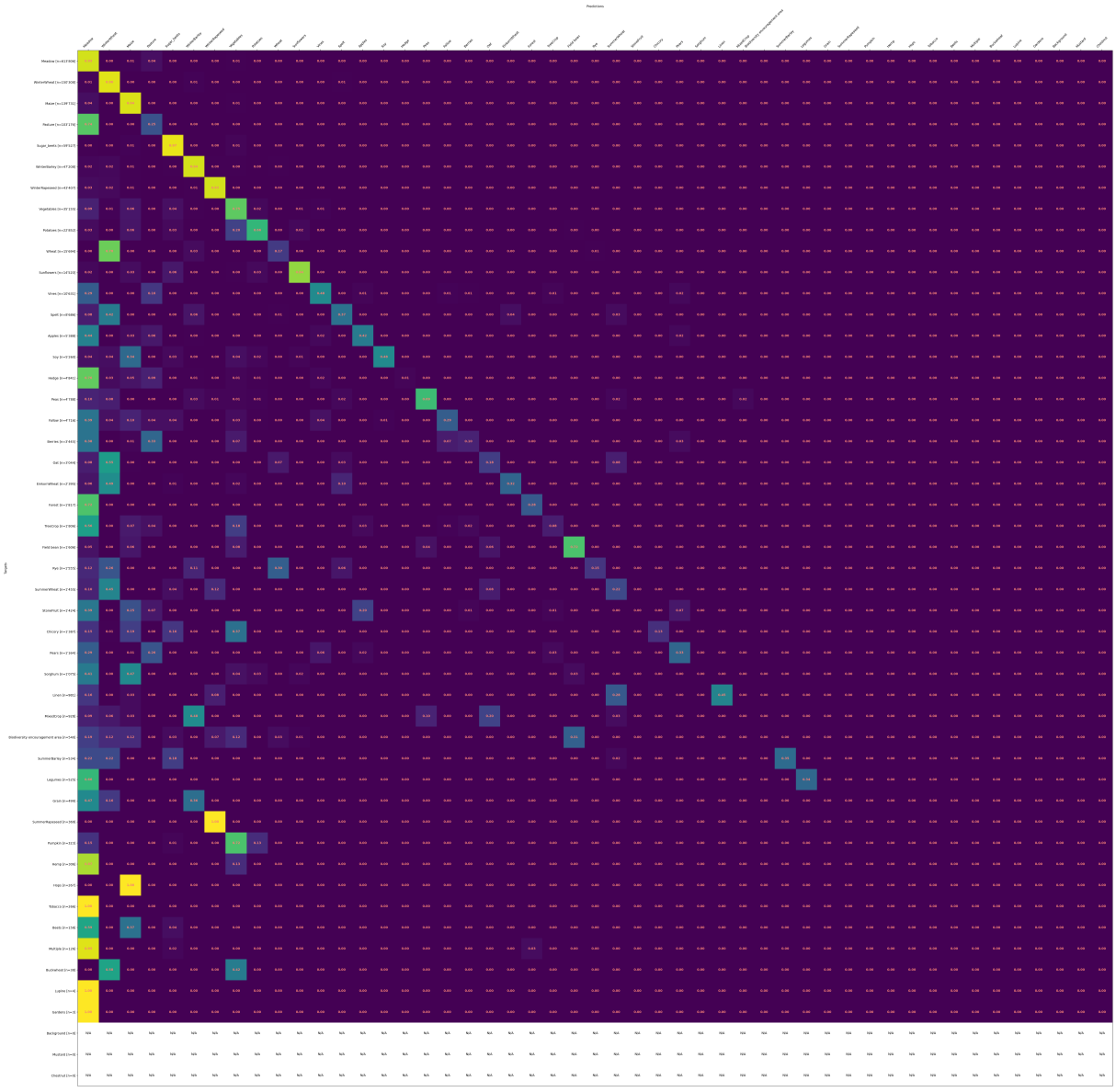


**Figure C.2:** Confusion matrix for best Messis model in Tier 3, calculated field majority on the sixth fold, our test fold.

# D Additional Experiments and Results

This appendix contains all experiments we did not deem relevant enough to be included in Section 5 Experiments and Results. They follow the same structure as the experiments included in the main part of the report.

## D.1 Tuning the Learning Rate

In this experiment, we evaluate the impact of different learning rates on the performance of Messis. We tune on the Adam optimizer, which has proven to be the best choice based on our previous Experiment 5.2 Comparing Optimizers. The learning rate is an important hyperparameter that controls the step size at each iteration while moving toward a minimum of the loss function. Tuning it can significantly affect the convergence speed and overall performance of the model. The results can be seen in Table D.1.

Experiment Setups:

- **adam-1e-3 (5-fold)**: Adam optimizer with a learning rate of $1 \times 10^{-3}$, no weight decay (baseline).
- **adam-1e-2 (5-fold)**: Adam optimizer with a learning rate of $1 \times 10^{-2}$, no weight decay.
- **adam-1e-4 (5-fold)**: Adam optimizer with a learning rate of $1 \times 10^{-4}$, no weight decay.

| Metric | † adam-1e-3 (5-fold) | adam-1e-2 (5-fold) | adam-1e-4 (5-fold) |
|---|---|---|---|
| F1 $T_1$ | **48.2% ± 4.7%** | 48.1% ± 5.9% | 45.0% ± 3.2% |
| F1 $T_2$ | 28.6% ± 0.8% | 28.7% ± 1.6% | **29.0% ± 1.3%** |
| F1 $T_3$ | 14.6% ± 0.3% | 14.5% ± 0.2% | **17.6% ± 1.6%** |
| Acc. $T_1$ | 46.4% ± 4.7% | **53.5% ± 7.1%** | 42.8% ± 3.0% |
| Acc. $T_2$ | 27.8% ± 0.7% | **30.2% ± 1.8%** | 27.8% ± 1.2% |
| Acc. $T_3$ | 15.0% ± 0.5% | 15.6% ± 0.6% | **17.3% ± 1.4%** |

**Table D.1:** Results for experiment exp-2-lr. All metrics calculated on field majority during validation phase, with mean and std reported. Best scores bold, second best underlined, † is baseline.

Key Findings:

- Baseline Performance (adam-1e-3): The baseline setup with a learning rate of $1 \times 10^{-3}$ showed strong performance in F1 scores for Tier 1 (48.2%) and Tier 2 (28.6%), with moderate accuracy across tiers.
- High Learning Rate (adam-1e-2): The setup with a learning rate of $1 \times 10^{-2}$ achieved the highest accuracy in Tier 1 (53.5%) and Tier 2 (30.2%), indicating faster convergence and better performance in terms of accuracy for higher tiers. However, it showed the lowest performance in Tier 3 F1 score (14.5%), possibly due to difficulty in capturing complex patterns for minority classes. Notably, the standard deviation in Tier 1 accuracy is unusually high and could indicate sensitivity to the learning rate.
- Low Learning Rate (adam-1e-4): The setup with a learning rate of $1 \times 10^{-4}$ achieved the highest F1 scores for Tier 2 (29.0%) and Tier 3 (17.6%), indicating better performance in capturing minority classes. It also showed the highest accuracy in Tier 3 (17.3%). However, it performed the worst on both metrics in Tier 1, especially accuracy (-10.7% against adam-1e-2). This suggests that a lower learning rate might slow down learning for more dominant classes.

Overall, lower learning rates seem to improve performance in minority classes, but slow down convergence for majority classes. Higher learning rates may lead to faster convergence, but potentially less stability and poorer performance in minority classes. Thus, the choice is a tradeoff between balanced results, good lower-tier scores and good higher-tier scores. The setup adam-1e-4 stands out as the best choice for Tier 3 performance, which we prioritize.

## D.2 Impact of Dropout Type and Probability

In this experiment, we evaluate the impact of switching the refinement head of Messis to Dropout2D and different dropout probabilities on the performance. Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of neurons to zero during training. This can improve the model's ability to generalize, as it forces the network to learn more robust features that are not dependent on the presence of any particular set of neurons (Srivastava et al., 2014). The results for the different dropout configurations are shown in Table D.2.

While all other head components of Messis use Dropout2D, the original refinement head proposed by Turkoglu et al. (2021), which we employ in Messis, uses Dropout1D with $p = 0.5$. This setup forms the baseline for this experiment. Dropout2D applies dropout to entire channels of the input tensor to subsequent convolutional layers, preserving spatial structure, while Dropout1D applies dropout independently to random elements within each channel.

The rationale for switching to Dropout2D in the convolutional refinement head too is that Dropout1D can disrupt spatial correlations, which has been shown to degrade performance. Tompson et al. (2015) found that Dropout1D does not properly regularize the activations in CNNs and mainly results in an effective learning rate decrease. By dropping entire channels, Dropout2D provides a more effective form of regularization for convolutional layers compared to Dropout1D, especially since the input comes from convolutional layers, where adjacent pixels within feature maps are typically strongly correlated.

Experiment Setups:

- **refhead-zuericrop (5-fold)**: Dropout1D in the refinement head with a dropout probability of 0.5 (baseline).
- **p-0.1 (5-fold)**: Dropout2D in all heads with a dropout probability of 0.1.
- **p-0.2 (5-fold)**: Dropout2D in all heads with a dropout probability of 0.2.
- **p-0.5 (1-fold)**: Dropout2D in all heads with a dropout probability of 0.5.

| Metric | † refhead-zuericrop (5-fold) | p-0.1 (5-fold) | p-0.2 (5-fold) | p-0.5 (1-fold) |
|---|---|---|---|---|
| F1 $T_1$ | **49.3% ± 5.9%** | 48.7% ± 5.0% | 48.5% ± 3.8% | 36.2% ± 0.0% |
| F1 $T_2$ | **31.8% ± 1.7%** | 31.5% ± 1.2% | 31.4% ± 1.0% | 27.8% ± 0.0% |
| F1 $T_3$ | 20.5% ± 0.6% | **20.5% ± 0.5%** | 19.9% ± 0.9% | 19.8% ± 0.0% |
| W. Acc. $T_1$ | 93.2% ± 0.2% | **93.3% ± 0.4%** | 93.2% ± 0.2% | 93.1% ± 0.0% |
| W. Acc. $T_2$ | 82.4% ± 0.4% | **82.6% ± 0.6%** | 82.4% ± 0.4% | 82.2% ± 0.0% |
| W. Acc. $T_3$ | 77.8% ± 0.9% | 78.0% ± 1.0% | 78.0% ± 0.5% | **78.8% ± 0.0%** |

**Table D.2:** Results for experiment exp-9-dropout2d. All metrics calculated on field majority during validation phase, with mean and std reported. Best scores bold, second best underlined, † is baseline.

Key findings:

1. **Baseline Performance:** The baseline setup (refhead-zuericrop) achieved the highest F1 scores for Tier 1 (49.3%) and Tier 2 (31.8%), which were better than those of other configurations. However, the baseline also exhibited higher std in some metrics, indicating less stability compared to other setups.
2. **Impact of Dropout2D with Lower Probabilities:** The p-0.1 setup showed competitive performance, with slightly higher weighted accuracy for Tier 1 (93.3%) and Tier 2 (82.6%), and relatively lower std values, indicating more stable and reliable performance. Its F1 scores were slightly lower than the baseline for Tier 1 and Tier 2, but still competitive.
3. **High Dropout Probability of 0.5:** Interestingly, the p-0.5 setup, with Dropout2D and a dropout probability of 0.5 (1-fold only), performed significantly worse in the F1 scores, especially at Tier 1 and 2. This indicates that while a high dropout rate can increase accuracy by regularizing the model, it may also hinder its ability to learn minority classes effectively. The model likely focuses more on the majority classes, which contributes to the highest Tier 3 weighted accuracy, at the expense of it's capability to capture finer details necessary for less represented classes, resulting in lower F1 performance. The per-class metrics logged to W&B of the fold-0 run of setup p-0.5 and setup p-0.1 support this interpretation.

The lower dropout probabilities with Dropout2D provide the expected balance between performance and stability. This supports the intuition that Dropout2D can be more effective for convolutional layers, especially with moderate dropout rates. The baseline (refhead-zuericrop) setup showed strong performance in F1 scores for Tier 1 and Tier 2, but with higher std values, indicating less consistency. Overall, the p-0.1 setup achieved the best balance between performance and stability. It provided the highest weighted accuracy in Tier 1 and Tier 2, while maintaining competitive F1 scores and lower std values.

## D.3 More Channels in Hierarchical Heads

In this experiment, we explore the impact of increasing the number of channels in the hierarchical head of our model. The standard setup reduces the embedding dimension of 2304 from the neck to 256 in the first layer of the head. We aimed to investigate whether this reduction might lead to a loss of important information to the model as more channels can carry more information.

Experiment Setups:

- **256 (1-fold)**: Utilizes 256 channels in all convolutional layers in the head part of our model (baseline).
- **1024 (1-fold)**: Utilizes 1024 channels in all convolutional layers in the head part of our model (baseline).

Due to resource constraints and a lower priority assigned to this experiment, we did not perform 5-fold cross-validation. Instead, we conducted a single-fold evaluation, making it challenging to compare the scores reported in Table D.3 directly. However, by comparing these results with mean standard deviations calculated from other experiments as described in Section 4.7.5 Estimation of Uncertainty in Evaluation and reported in Table 4.1, we can infer whether using 256 or 1024 channels offers any significant advantage.

Key Findings:

- **Baseline Performance (256)**: This configuration achieved a higher F1 score for Tier 1 (41.6%) compared to the 1024 channels setup.
- **More channels (1024)**: This configuration outperformed the 256 channels setup in F1 score for Tier 2 (31.3%) and Tier 3 (21.1%) and weighted accuracy across all tiers: W. Acc. $T_1$ (93.2%), W. Acc. $T_2$ (81.7%), and W. Acc. $T_3$ (78.0%).

| Metric | † 256 (1-fold) | 1024 (1-fold) |
|---|---|---|
| F1 $T_1$ | **41.6% ± 0.0%** | 41.5% ± 0.0% |
| F1 $T_2$ | 30.4% ± 0.0% | **31.3% ± 0.0%** |
| F1 $T_3$ | 19.9% ± 0.0% | **21.1% ± 0.0%** |
| W. Acc. $T_1$ | 93.1% ± 0.0% | **93.2% ± 0.0%** |
| W. Acc. $T_2$ | 81.0% ± 0.0% | **81.7% ± 0.0%** |
| W. Acc. $T_3$ | 76.8% ± 0.0% | **78.0% ± 0.0%** |

**Table D.3:** Results for experiment exp-11-more-channels. All metrics calculated on field majority during validation phase, with mean and std reported. Best scores bold, second best underlined, † is baseline.

Comparing the score differences with the mean standard deviations from Table 4.1, we can conclude that the 1024 setup outperformed the 256 setup by more than one standard deviation on Tier 3 for both the F1 and the weighted accuracy score. These differences are significant for the two Tier 3 scores and do suggest a slight but significant advantage when using more channels, especially at the most granular level.

## D.4 More Convolutional Layers in Hierarchical Heads

In this experiment, we evaluate the impact of adding more convolutional layers within each module of the head component of our model. Currently, each module between a tier in the head component uses a single convolutional layer. Inspired by state-of-the-art models that classify images, such as AlexNet, which are deep neural networks with many layers, we wanted to investigate whether increasing the number of convolutional layers would enhance the performance of our model.

As mentioned in Experiment D.3 More Channels in Hierarchical Heads we were limited by resources, so we did not perform 5-fold cross-validation to estimate the variability but used the standard deviations reported in Table 4.1 to perform a meaningful comparison. The 1-fold scores are reported in Table D.4.

Experiment Setups:

- **1 (1-fold)**: The baseline configuration with one convolutional layer per module in the head.
- **3 (1-fold)**: A configuration with three convolutional layers per module in the head.
- **5 (1-fold)**: A configuration with five convolutional layers per module in the head.

| Metric | † 1 (1-fold) | 3 (1-fold) | 5 (1-fold) |
|---|---|---|---|
| F1 $T_1$ | **41.7% ± 0.0%** | 37.3% ± 0.0% | 36.9% ± 0.0% |
| F1 $T_2$ | **29.8% ± 0.0%** | 28.3% ± 0.0% | 28.5% ± 0.0% |
| F1 $T_3$ | 17.9% ± 0.0% | **20.8% ± 0.0%** | 19.2% ± 0.0% |
| W. Acc. $T_1$ | 92.8% ± 0.0% | **92.8% ± 0.0%** | 92.5% ± 0.0% |
| W. Acc. $T_2$ | **81.9% ± 0.0%** | 79.7% ± 0.0% | 79.8% ± 0.0% |
| W. Acc. $T_3$ | **78.2% ± 0.0%** | 75.8% ± 0.0% | 74.1% ± 0.0% |

**Table D.4:** Results for experiment exp-12-num_convs. All metrics calculated on field majority during validation phase, with mean and std reported. Best scores bold, second best underlined, † is baseline.

Key Findings:

- **Baseline Performance (1 layer)**: The baseline configuration with one convolutional layer per module achieved the highest scores for 4 out of the 6 reported metrics, with F1 scores of 41.7%, 29.8%, for $T_1$ and $T_2$ respectively and weighted accuracies of 81.9% and 78.2% for $T_2$ and $T_3$ respectively.
- **Adding Layers (3 layers)**: The configuration with three convolutional layers per module had the best F1 score for $T_3$ of 20.8% and the same weighted accuracy score on $T_1$ as the baseline.
- **Increasing Layers Further (5 layers)**: The configuration with five convolutional layers per module performed the worst overall but still had the second-best score in 3 metrics.

The only differences between scores greater than one standard deviation from Table 4.1 were: the F1 $T_1$ score for the 3 layers setup and the weighted accuracies on $T_2$ and $T_3$ for the one-layer setup. This suggests that adding more convolutional layers in the head component of our model does not improve performance and may degrade it. This further indicates that for our model, increasing the depth of the head component does not provide additional benefits.

## D.5 Further Experiment Ideas

We had ideas for several additional experiments that did not make it into the prioritization. They are listed below, and could, when performed, provide further relevant insights into the behavior, strengths, and weaknesses of Messis.

1. Temporal Resolution:
   - **Composite Satellite Images** - As an alternative to increasing timesteps, create composite images along the temporal axis, e.g., aggregate 14 days using mean/median.
   - **Ensemble Model** - Train season-specific crop classification models and combine their results for predictions.
2. Effect of Hierarchy:
   - **Compare Tier 3 Refined Derived Predictions against Model Predictions for all Tiers 1, 2, 3** - Calculate metrics for all tiers based on their specific hierarchical heads prediction, ignoring the refinement head. Log both to compare performance of the derived predictions for Tier 1 and 2 against the hierarchical heads prediction.
3. Miscellaneous:
   - **Use More Satellite Bands** - Use additional satellite bands beyond the current 6, as suggested by other relevant papers.
   - **Model Architecture with Parallel instead of Serial Classification** - Predict tiers 1, 2, and 3 in parallel on neck output, then concatenate, instead of passing transformed features sequentially from tier 1 to tier 2, etc.
   - **Class-Based Weighted Loss Penalty** - Apply class-based weighted loss penalties based on the inverse proportional share of the classes in the dataset. Consider whether false positives or false negatives are more critical for the model's application, as such penalties can affect majority class performance. Keep in mind the main crops in Switzerland: maize, barley, wheat, potatoes, sugar beets, rapeseed etc.