

Knowledge-Driven Intelligent Systems for Precision Agriculture

A Demonstration on Disease Spread Prediction

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Maurice Günder

aus

Heinsberg

Bonn, im März 2026

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter/Betreuer: Prof. Dr. Christian Bauckhage
Gutachter: Prof. Dr. Stefan Wrobel

Tag der Promotion: 23. Juni 2026
Erscheinungsjahr: 2026

Abstract

Organic cultivation, resource efficiency, and sustainability are gaining increasing importance in times of climate change. To address these issues, modern agriculture is increasingly relying on the use of drones to monitor and analyze crop stands. They enable non-invasive and rapid collection of large-scale imagery and geodata. A logical step in this development is the processing of these data through the integration of artificial intelligence (AI). Purely data-driven approaches, however, quickly reach their limits when only small amounts of annotated data are available or the collection of additional (meta) data is associated with high personnel and financial effort.

In agriculture, decades of research have already produced extensive expert knowledge in a wide range of disciplines. This knowledge can be used to improve the performance of data-driven AI models and increase their applicability in practice. This dissertation aims to show how data-driven models can be combined with knowledge in order to address important agricultural challenges, while at the same time remaining generally applicable.

As a use case, the spread of the leaf spot disease *Cercospora* in sugar beet is examined, from data collection with drones, through the prediction of infestation severity, to epidemiological modeling. The first building block is the acquisition of an extensive image database from the georeferenced drone recordings. Using robust algorithms, a semi-automated workflow is developed to detect plant positions in temporally separated drone images of sugar beet fields. The result is a spatially and temporally referenced database of individual plant images, which serves as the basis for the second building block. In this second step, the image data is used, together with environmental data and assessments by human experts, to train an AI model to predict the severity of *Cercospora* infestation. The third building block uses this model to estimate the time of infection of individual plants. In combination with meteorological data and epidemiological knowledge, an infection model is developed that can predict the spatio-temporal spread of *Cercospora* in sugar beet stands. This could enable the agriculture of the future to take targeted and needs-based measures to combat plant diseases and thus minimize the use of plant protection products.

The AI models developed are in many respects based on the transformer architecture, which in recent years has achieved partly disruptive advances in various domains. When used deliberately, they enable high interpretability and realistic modeling of the underlying processes. A particular focus of this dissertation is on model interpretability. Through application-oriented modeling, models can be designed in such a way that their predictions are comprehensible to domain experts. This is an important step toward increasing trust in AI-based systems and promoting their acceptance in practice.

Zusammenfassung

Ökologischer Anbau, Ressourceneffizienz und Nachhaltigkeit gewinnen in Zeiten des Klimawandels immer mehr Bedeutung. Um diese Themen zu adressieren, setzt die moderne Landwirtschaft zunehmend auf den Einsatz von Drohnen zur Überwachung und Analyse von Nutzpflanzenbeständen. Sie ermöglichen eine nichtinvasive und schnelle Erhebung großflächiger Bild- und Geodaten. Ein logischer Schritt in dieser Entwicklung ist die Verarbeitung dieser Daten durch Integration moderner Methoden der Künstlichen Intelligenz (KI). Rein datengetriebene Ansätze stoßen dabei jedoch schnell an ihre Grenzen, wenn nur limitierte Mengen annotierter Daten vorliegen oder die Erhebung weiterer (Meta-)Daten mit hohem personellen und finanziellen Aufwand verbunden ist.

In der Landwirtschaft ist durch jahrzehntelange Forschung bereits umfangreiches Expertenwissen in verschiedensten Disziplinen verfügbar. Dieses Wissen kann genutzt werden, um die Leistung von datengetriebenen KI-Modellen zu verbessern und deren Anwendbarkeit in der Praxis zu erhöhen. In dieser Dissertation soll gezeigt werden, wie datengetriebene Modelle mit Wissen kombiniert werden können, um wichtige landwirtschaftliche Herausforderungen zu adressieren und gleichzeitig generalistisch einsetzbar zu sein.

Als Anwendungsfall wird die Ausbreitung der Blattfleckenkrankheit *Cercospora* in Zuckerrüben von der Datenerhebung mit Drohnen über die Vorhersage der Befallsstärke bis hin zur epidemiologischen Modellierung betrachtet. Der erste Baustein ist die Akquise einer umfangreichen Bilddatenbank aus den georeferenzierten Drohnenaufnahmen. Mittels robuster Algorithmen wird ein teil-automatischer Workflow zur Erkennung von Pflanzenpositionen in zeitlich getrennten Drohnenaufnahmen von Zuckerrübenfeldern entwickelt. Das Ergebnis ist eine räumlich und zeitlich referenzierte Datenbank einzelner Pflanzenbilder, die als Grundlage für den zweiten Baustein dient. In diesem werden die Bilddaten verwendet, um mit Hilfe von Umweltdaten und menschlichen Expertenbonitoren ein KI-Modell zur Vorhersage der *Cercospora*-Befallsstärke zu trainieren. Im dritten Baustein wird dieses Modell wiederum verwendet, um den Infektionszeitpunkt individueller Pflanzen zu schätzen. In Verbindung mit meteorologischen Daten und epidemiologischem Wissen wird ein Infektionsmodell entwickelt, das die räumlich-zeitliche Ausbreitung von *Cercospora* in Zuckerrübenbeständen vorhersagen kann. Dies könnte der Landwirtschaft der Zukunft ermöglichen, gezielte und bedarfsgerechte Maßnahmen zur Bekämpfung von Pflanzenkrankheiten zu ergreifen und so den Einsatz von Pflanzenschutzmitteln zu minimieren.

Die entwickelten KI-Modelle orientieren sich in vielen Aspekten an der Transformer-Architektur, die in den letzten Jahren in verschiedensten Domänen teils disruptive Fortschritte erzielt hat. Bewusst eingesetzt ermöglichen sie eine hohe Interpretierbarkeit und realitätsnahe Modellierung der zugrundeliegenden Prozesse. Insbesondere auf der Modellinterpretierbarkeit liegt ein Fokus in dieser Dissertation. Durch anwendungsnahe

Modellierung können Modelle gestaltet werden, deren Vorhersagen für Domänenexperten nachvollziehbar sind. Dies ist ein wichtiger Schritt, um das Vertrauen in KI-gestützte Systeme zu erhöhen und deren Akzeptanz in der Praxis zu fördern.

Contents

1	Introduction	1
1.1	Precision Agriculture	1
1.1.1	Remote Sensing in Precision Agriculture	2
1.1.2	Machine Learning in Precision Agriculture	4
1.2	Attention Mechanisms and Transformers	5
1.2.1	General Formulation	6
1.2.2	Scaled Dot-Product Attention	7
1.2.3	Self-Attention as Contextualization	8
1.2.4	Multi-Head Attention and the Transformer Block	8
1.2.5	Interpretability of Attention and Transformers	9
1.2.5.1	Direct Inspection of Attention Weights	9
1.2.5.2	Limitations: Is Attention an Explanation?	10
1.2.5.3	Aggregated and Layerwise Analyses	10
1.2.6	Attention as a Tool for Model Inspection	11
1.3	Incorporating Knowledge into Data-Driven Models	11
1.4	Goal and Structure	13
1.4.1	Cercospora Leaf Spot in Sugar Beet	14
1.4.2	Disease Severity Estimation	15
1.5	Data Basis	17
2	Plant Cataloging	21
2.1	Introduction	21
2.2	Background	23
2.3	Data Description	24
2.4	Plant Extraction Workflow	25
2.4.1	Preprocessing	25
2.4.2	Plant Position Detection	25
2.4.2.1	Vegetation Index	26
2.4.2.2	Plant-soil Segmentation	27
2.4.2.3	Growth Function	30
2.4.2.4	Filtering and Peak Finding	31
2.4.3	Plant Group Registration	34
2.4.3.1	Aligning Plant Positions	34
2.4.3.2	Recognizing Seeding Lines	37
2.4.3.3	Filtering Weed	42
2.4.3.4	Linking Individual Plants during Time	43
2.4.3.5	Further Filtering and Indirect Detections	46
2.5	Evaluation and Discussion	47

2.5.1	Application to In-Field Annotations	51
2.6	Interim Conclusion	51
3	Disease Severity Prediction	54
3.1	Introduction	54
3.2	Data and Preprocessing	56
3.2.1	Using Weather Data for Environmental Features	57
3.2.2	Image Normalization	58
3.2.3	Data Augmentation	61
3.3	Deep Label Distribution Learning	61
3.3.1	Full Kullback-Leibler Divergence Loss	62
3.3.2	Multi-Head Regression	64
3.4	SugarViT Architecture	64
3.4.1	Vision Transformer Backbone	66
3.4.2	MLP Neck	66
3.4.3	LDL Heads	67
3.4.4	Feature Mixing	67
3.5	Results	68
3.5.1	Standardization vs. Histogram Equalization	71
3.5.2	SugarViT Pretraining	72
3.5.3	Comparison Experiments	77
3.5.3.1	Backbone Network	77
3.5.3.2	Pretraining	78
3.5.4	Evaluation on the Test Dataset	81
3.5.5	Application in the Field	85
3.5.6	Attention Maps	86
3.6	Discussion	89
3.7	Interim Conclusion	90
4	Disease Spread Modeling	92
4.1	Introduction	92
4.2	Plant-specific Parameters from Empirical Knowledge	94
4.3	Estimation of Infection Date	98
4.4	Gathering the Infection Data	100
4.5	Agent-based Infection Simulation Model	101
4.6	Infection Hazard Modeling	102
4.7	SpreadNet	104
4.7.1	Data Preprocessing and Augmentation	104
4.7.2	SpreadNet Architecture	107
4.7.2.1	Hourly Temporal Encoding	109
4.7.2.2	Cross-Day Self-Attention Pooling	109
4.7.2.3	Fusion of Ego-Plant Features and Daily Time Encoding	110
4.7.2.4	Ego-conditioned Neighbor Attention	111
4.7.2.5	Ego-Neighbor Fusion	113
4.7.2.6	Daily Transformer Encoder	114
4.7.2.7	Hazard Head and Daily Infection Hazards	114
4.7.3	Hybrid Nowcasting Survival Loss Function	116
4.7.4	Sampling Strategy	117

4.7.5	Evaluation Metrics	118
4.7.5.1	Nowcasting quality (Objective O1)	118
4.7.5.2	Infection-time prediction (Objective O2)	119
4.7.6	Model Training	122
4.7.7	Model Application during Simulation	124
4.8	Results	125
4.8.1	Evaluation on the Datasets	125
4.8.1.1	Metrics	125
4.8.1.2	Saliency and Attention Maps	128
4.8.2	Evaluation on the Simulation	134
4.9	Discussion	139
4.10	Interim Conclusion	140
5	Conclusion	142
5.1	Key Contributions	142
5.1.1	Plant Cataloging	142
5.1.2	Disease Severity Prediction	143
5.1.3	Disease Spread Modeling	144
5.1.4	End-to-End Workflow	145
5.2	Limitations	145
5.2.1	Methodology	145
5.2.2	Data Availability	147
5.3	Future Work	148
5.3.1	Technical and Scientific Perspective	148
5.3.2	Socio-Economic Perspective	150
	List of Acronyms	I
	Own Publications	IV
	References	V

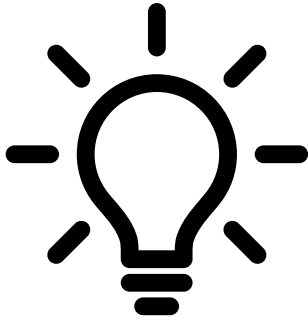
List of Figures

1.1	Main building blocks of this dissertation	14
1.2	Symptoms of CLS on a sugar beet leaf	15
1.3	Used DS scale for the prediction model with example images	16
2.1	Overview of plant cataloging and image extraction workflow	22
2.2	Overview of CLS experimental field 2020	24
2.3	Different VIs for discrimination between plants and soil	27
2.4	Plant-soil segmentation for different growing stages	29
2.5	Growth function curve fits	30
2.6	Plant peaks for different growing stages	33
2.7	Point cloud alignment	37
2.8	Seeding line rotation angle determination with Hough transform	38
2.9	Seeding line recognition	39
2.10	Weed filtering	42
2.11	Cluster centroids	43
2.12	Group label sorting	46
2.13	Precision and recall evaluation summary	50
2.14	Plant catalog for in-field observations with QField	51
2.15	Plant image time series	53
3.1	Conceptual overview of the disease severity prediction chapter	55
3.2	Histograms of available labels for DS, NPG, and GDD	58
3.3	Example images	60
3.4	Sketch of SugarViT	65
3.5	Output of SugarViT	70
3.6	Results of the standardization vs. histogram equalization experiment	72
3.7	Results of the SugarViT pretraining	75
3.8	Validation MDO by training epoch of the SugarViT pretraining for channel dropout and RGB-only	76
3.9	Training loss components by training epoch of the SugarViT DS training for channel dropout and RGB-only	76
3.10	Comparison of backbone networks	78
3.11	Comparison of the pretrained and non-pretrained SugarViT model	80
3.12	Exemplary application of SugarViT for disease severity prediction on unseen UAV data	85
3.13	Attention maps for an example image per DS class	88
4.1	Comparison between JDN and the GDD and ETo used as “phenological time”	95

4.2	Relations of cover ratio, plant radius, and leaf stage versus GDD	96
4.3	Functional relations between temperature, relative humidity, and disease development	97
4.4	Exemplary visualization for the retrieval of the infection date	98
4.5	Infection map for the D06 dataset	99
4.6	Generic interaction principle of the modeling approach	101
4.7	Overview of the SpreadNet architecture.	108
4.8	MAE for event day vs. data window length	127
4.9	Input saliencies for all features	131
4.10	Attention weights for the cross-day self-attention pooling module	132
4.11	Attention weights for ego-conditioned neighbor attention module and mean neighbor distance	132
4.12	Cumulative infected plants over the harvest cycle	133
4.13	New infections per day for all datasets	134
4.14	Target-day autonomy analysis summary	137
4.15	Target-day autonomy analysis heatmaps	138

List of Tables

1.1	Overview of recorded environmental data used for the sugar beet field experiments	19
1.2	Spectral ranges of the two used camera sensor systems	19
1.3	Overview of datasets used for disease severity estimation and spread modeling	20
3.1	Model configuration for comparison between standardization and histogram equalization	71
3.2	SugarViT model configuration for pretraining and finetuning	74
3.3	Evaluation metrics for the model with best validation MDO on the test dataset separated by true DS value	83
3.4	Evaluation metrics for the model with best validation loss on the test dataset separated by true DS value	84
4.1	Plant parameters structured by non-plant-specific, plant-specific and disease-specific variables	98
4.2	Data augmentation parameters for SpreadNet	106
4.3	SpreadNet model configurations	123
4.4	Performance metrics for SpreadNet variants	126



1

Introduction

As an initial step, this chapter provides necessary background information on the key concepts and technologies that underpin the research presented in this dissertation. Firstly, the idea and potential of precision agriculture is introduced, particularly focusing on remote sensing (RS) and machine learning (ML) in this context. Secondly, a brief deep dive into the attention mechanism and the transformer architecture is given, and it is explained how these can be used to incorporate knowledge into ML models. Subsequently, other concepts of knowledge integration are briefly introduced. Finally, the research questions of this dissertation are motivated, formulated, and structured. After an outline of the dissertation's structure, the chapter concludes with a description of the underlying data that will be used throughout the research.

1.1 Precision Agriculture

Precision agriculture is an agricultural management concept that uses modern information and communication technologies to increase crop yields, optimize the use of resources, and reduce environmental impact. It emerged in the mid-1980s with location-specific crop production research investigating which treatments should be carried out at which location and time. [91, 112, 166] Connected with the development of geographic information systems (GIS) and global positioning system (GPS) technologies, precision

agriculture gained more and more attention and importance in modern farming practices throughout the last decades by enhancing yield monitoring, remote and proximal sensing, and variable-rate application technologies. Thus, it enables farmers to measure, map, and manage heterogeneous conditions regarding soil, crop, and environmental conditions in their fields. [39, 97]

Especially in times of climate change, growing world population, and, thus, increasing food demand, precision agriculture is considered a key technology to ensure global food security while minimizing environmental impact. [36, 43] By targeting inputs such as seed, fertilizers, and agrochemicals to the specific needs of crops at granular spatial scales, it can increase resource efficiency, reduce environmental externalities such as nutrient deficiencies or greenhouse gas emissions, and support regulatory compliance [19, 39]. At the same time, the integration of digital technologies, including the internet of things (IoT), cloud computing, and data analytics, is transforming precision agriculture into “smart farming”, where large volumes of data from sensors, machinery, and satellites are translated into actionable decision support for farmers and advisors [158].

The technological success of precision agriculture is unquestionable. However, its widespread adoption is still in progress. There are large differences in the level of adoption between regions, farm sizes, and production systems due to various factors such as investment costs, skills and advisory services, data governance, and the compatibility of technologies with existing farm operations [129, 166]. Ongoing research therefore focuses not only on the algorithmic and technological improvements, but also on understanding socioeconomic barriers, demonstrating economic returns, and designing policies and business models that can support broader and more inclusive uptake. In this context, precision agriculture is increasingly framed as a critical lever for achieving both productivity and sustainability goals in modern agriculture.

Still, this dissertation will focus on the technological side with two main aspects of precision agriculture: RS and the use of ML techniques for data analysis and modeling. Therefore, these two aspects will be particularly introduced in the following sections.

1.1.1 Remote Sensing in Precision Agriculture

Remote sensing (RS) enables non-invasive, large-scale, and cost-efficient data acquisition of agricultural fields and, therefore, is a key technology for the success of precision agriculture. Various platforms for RS exist, ranging from satellite-based systems over manned aircraft, unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) to stationary sensors. [121] Satellite missions such as Landsat, Sentinel, and commercial constellations provide data from emitted or reflected radiation in multiple spectral bands. This so-called multispectral imaging on regular flight missions provides repeated coverage suitable for monitoring crop development, mapping yield-limiting

zones, and supporting variable-rate application maps at the field to a regional level [8, 96, 97]. For investigations on individual field levels, UAV-borne sensors like drones offer much finer spatial resolutions and on-demand acquisitions, enabling detailed scouting for nutrient deficiencies, lodging, weeds, and localized disease outbreaks [165]. Unlike satellite-based RS, the UAV-based RS offers low-cost solutions for private usage and individual applications. There are many conceivable use cases like plant species segmentation [62], multi-sensor plant analyses [49, 58, 128], automated plant counting [142], general phenotyping [25, 160], or health status analyses [10, 23, 24, 159]. Among those, several tasks are based on investigations on individual plants. This is challenging for applications in real-world farming where the fields are densely seeded, in contrast to research cases where plants could possibly have larger distances.

Beyond traditional multispectral imagery, hyperspectral sensors, long-wave (thermal) infrared (LWIR), synthetic aperture radar (SAR), and light detection and ranging (LiDAR) extend the observable crop traits to biochemical composition, canopy temperature—as a proxy for water stress—and structural parameters, respectively [8, 87, 140]. Furthermore, vegetation indices (VIs) such as the normalized difference vegetation index (NDVI) are advanced spectral indicators that are used to derive maps of biomass, leaf area index, and chlorophyll content that can inform nutrient management and yield prediction [59, 161]

Despite these advances, several challenges still constrain the full exploitation of remote sensing for precision agriculture. Trade-offs among spatial resolution, temporal frequency, and cost, as well as issues such as cloud cover, sensor calibration, and the need for robust ground truth, can limit operational reliability [96, 97]. Moreover, translating imagery into actionable, farm-specific recommendations requires expertise, scalable processing pipelines, and business models that work for farms of different sizes and contexts. Thus, ongoing research focuses on data fusion, e.g., combining satellite, UAV, and proximal data [149], new hyperspectral and radar satellite missions [20, 138], and user-friendly decision support tools [93], positioning remote sensing as a central component of increasingly data-driven, “smart” precision agriculture systems.

This dissertation will make use of the cost-effective UAV imaging which enables having multiple images during a complete growing season and combine this information for localizing plants in a spatio-temporal way. Reasons and use cases for cataloging and detecting plants in the field are manifold. On the one hand, it allows determining the number of plants in the field accurately. On the other hand, it enables the extraction of traits of great interest, such as the distance between plants and plant density. Individualization and identification of each plant in the field and the opportunity of retrieving individual plants again in a time series allow generating a vast database of annotations. The availability of this data framework enables the training more robust machine learning models for disease severity analysis, to mention just one possible use case. In the

following section, the state of the art regarding ML methods in precision agriculture is briefly introduced.

1.1.2 Machine Learning in Precision Agriculture

The particular benefit of RS in precision agriculture lies in its ability to capture spatial and temporal data from a large variety of sensors. This rich, multimodal data can be leveraged by ML techniques to extract patterns and generate predictions that were not visible or accessible with traditional methods like mechanistic modeling or rule-based methods alone. As a result, the value of RS in precision agriculture increasingly lies in its integration with other data layers and digital technologies. When combined with soil maps, yield monitor data, proximal sensors, and crop growth models in a GIS, remotely sensed information helps delineate management zones, calibrate and validate models, and refine site-specific recommendations [39, 97]. Analyzing and interpreting resulting data with ML and pattern recognition methods has the potential to gain economical and ecological efficiency, which is why computational intelligence and artificial intelligence (AI) is increasingly applied in agricultural contexts [11, 67, 78]. Furthermore, recent advances in cloud computing, and the availability of open-access datasets (e.g. Sentinel, Landsat) are enabling near-real-time services for in-season yield forecasting, nitrogen management, irrigation scheduling, and risk assessment at operational scales [8, 80].

ML has developed as a key driver for precision—and especially “smart”—agriculture, enabling data-driven decision-making across the entire agricultural value chain. It has the capability to learn patterns from heterogeneous data domains, from proximal and remote sensing to machine logs, weather records and forecasts, as well as economic information. Thus, ML models can support more precise, timely, and site-specific management than rule-based systems or traditional statistics alone [78, 158]. Advances in deep learning—supported by increased computational power and the evolution of sensing technologies—now allow complex nonlinear relationships between environment, management, and crop performance to be captured and learned at operational scales. [26, 64, 68, 76, 77, 106]

Applications of ML in agriculture incorporate yield prediction, crop and soil monitoring, disease and weed detection, phenotyping, and farm automation. Convolutional neural networks (CNNs) and related deep learning architectures are widely used to classify crops, detect stresses, and segment weeds and disease symptoms from UAV and satellite imagery, as well as from ground-based RGB and hyperspectral images [67, 78]. Supervised learning and ensemble methods, e.g. random forests and gradient boosting, support many state-of-the-art approaches to crop yield prediction and nitrogen status estimation, often outperforming classical regression methods when trained on multi-year, multi-location datasets [26, 143]. Simultaneously, ML is increasingly embedded in

robotic platforms for autonomous navigation, precision spraying, harvesting, and intra-field decision support [64].

The effectiveness of ML in agriculture critically depends on integrating diverse data sources in both space and time. Multi-sensor and multi-scale “big data” workflows combine satellite and UAV imagery, in-situ sensor data, weather forecasts, and farm management records to develop predictive and prescriptive models for irrigation scheduling, fertilization, and risk management [68, 158]. Deep learning models are being tailored for spatio-temporal agricultural data, for example by combining CNNs with recurrent or attention-based architectures to exploit time-series imagery and weather information for in-season yield forecasting and early-warning systems [78, 143]. These developments position ML as a central component of integrated decision-support systems for precision agriculture and broader sustainability and climate-adaptation strategies.

Despite rapid progress, several challenges, however, limit the full deployment of ML in agriculture. Robust model development is constrained by data quality, label scarcity, and the difficulty of capturing management practices and socio-economic factors at scale [64, 78]. Generalization across regions, seasons, and cultivars remains a major issue, often requiring transfer learning, domain adaptation, or site-specific recalibration. Additionally, concerns about model interpretability, data ownership, and access to digital tools are increasingly prominent in the design and governance of ML-based agricultural services [68, 158]. Addressing these challenges is crucial to moving from proof-of-concept studies toward trustworthy, scalable ML solutions, which both large- and small-scale farming systems can benefit from.

In the scope of research presented in this dissertation, several ML driven approaches were conducted. In [I], an image-to-image translation approach based on a U-Net [123] was developed to translate UAV images into the domain of satellite images. Moreover, first steps toward hybrid modeling approaches combining mechanistic models with ML techniques were taken [H]. Besides image-based ML approaches, also the field of genetic or evolutionary algorithms was explored for optimizing cropping strategies [D]. In this work, a multi-objective optimization heuristic was developed to optimize the harvest scheduling of crops under various boundary conditions like uncertain weather or logistical limits. Furthermore, in the field of active learning, novel methods for data-efficient learning with near infrared (NIR) spectroscopy were proposed [J].

1.2 Attention Mechanisms and Transformers

Since the attention mechanism plays a major role in the modeling approaches throughout this dissertation, a brief overview of attention and its mathematical formulation is given in the following.

The attention mechanism has become a central building block of modern deep learning architectures, especially in natural language processing and time series modeling. Originally introduced to augment recurrent neural networks (RNNs) in neural machine translation [9, 85], attention enables a model to dynamically focus on different parts of its input when producing each output element. The transformer architecture [144] replaces recurrence with stacked layers of self-attention and conventional feed-forward networks, achieving state-of-the-art performance in many domains.

Moreover, attention is particularly attractive because it exposes *attention weights* that can be inspected and visualized. These weights are often interpreted as an insight into the model’s internal reasoning process. However, the opinion “attention is explanation” is controversially discussed in the literature. [61, 130, 150]. The research presented in this dissertation will, nevertheless, shed light on how attention can at least help to understand the model’s behavior and suggest if the model is using the right information for its predictions.

In this section, the attention mechanism and the transformer’s so-called multi-head self-attention is described mathematically. Afterward, it is briefly discussed how and to what extent these mechanisms can be used for model interpretability.

1.2.1 General Formulation

Conceptually, an attention mechanism maps a *query* vector to a weighted combination of a set of *value* vectors, where the weights are determined by similarity between the query and a corresponding set of *key* vectors [144]. Formally,

$$\mathbf{Q} \in \mathbb{R}^{n_q \times d_k}, \quad \mathbf{K} \in \mathbb{R}^{n_k \times d_k}, \quad \mathbf{V} \in \mathbb{R}^{n_k \times d_v} \quad (1.1)$$

denote matrices of queries, keys, and values, respectively. Here, n_q is the number of queries, n_k the number of keys (and values), and d_k, d_v their feature dimensions.

For a single query vector $q_i \in \mathbb{R}^{d_k}$ (the i -th row of $\mathbf{Q}$), the attention mechanism computes

$$e_{ij} = a(q_i, k_j), \quad j \in \{1, \dots, n_k\}, \quad (1.2)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{l=1}^{n_k} \exp(e_{il})}, \quad (1.3)$$

$$\text{Attention}(q_i, \mathbf{K}, \mathbf{V}) = \sum_{j=1}^{n_k} \alpha_{ij} v_j, \quad (1.4)$$

where $a(\cdot, \cdot)$ is some *compatibility* or *score* function, e_{ij} are unnormalized scores, α_{ij} are attention weights, and $v_j \in \mathbb{R}^{d_v}$ is the j -th row of $\mathbf{V}$.

The softmax function in (1.3) ensures that the attention weights are non-negative and normalized. Hence,

$$\alpha_{ij} \geq 0, \quad \sum_{j=1}^{n_k} \alpha_{ij} = 1, \quad (1.5)$$

so that each row $\alpha_i = (\alpha_{i1}, \dots, \alpha_{in_k})$ can be interpreted as a probability distribution over the keys (and corresponding values) conditioned on the query q_i . This probabilistic interpretation is central to many interpretability analyses: inspecting α_{ij} shows where the model is “attending” when computing the representation for q_i .

1.2.2 Scaled Dot-Product Attention

The transformer architecture uses *scaled dot-product attention* [144], where the compatibility function is a scaled inner product:

$$a(q_i, k_j) = \frac{q_i k_j^\top}{\sqrt{d_k}}. \quad (1.6)$$

Stacking all queries and keys, the matrix formulation is:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} \right) \mathbf{V}, \quad (1.7)$$

where the softmax is applied row-wise.

The matrix $\mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{n_q \times n_k}$ contains pairwise similarity scores between queries and keys; the normalized matrix

$$\mathbf{A} = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} \right) \in \mathbb{R}^{n_q \times n_k} \quad (1.8)$$

collects the attention weights, with each row $\mathbf{A}_{i,:}$ being a probability distribution over the n_k keys for query i . The output is then $\mathbf{A}\mathbf{V} \in \mathbb{R}^{n_q \times d_v}$, i.e., each query is mapped to a convex combination—due to the softmax characteristics—of value vectors.

Geometrically, the dot product $q_i k_j^\top$ measures similarity in the shared feature space. The softmax emphasizes keys with high similarity and suppresses others. The scaling factor $1/\sqrt{d_k}$ prevents the dot products from growing too large in magnitude as d_k increases, which would otherwise push the softmax into regions with vanishing gradients [144].

In transformers, the queries, keys, and values are obtained by applying learned linear projections to the same input sequence $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$, so that

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \quad (1.9)$$

with $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $\mathbf{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_v}$.

1.2.3 Self-Attention as Contextualization

In *self-attention*, the sequence serves simultaneously as queries, keys, and values. For a sequence of input embeddings

$$\mathbf{X} = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d_{\text{model}}}, \quad (1.10)$$

the self-attention output is

$$\text{SelfAttn}(\mathbf{X}) = \text{Attention}(\mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V). \quad (1.11)$$

Thus, the i -th row of $\text{SelfAttn}(\mathbf{X})$ is

$$y_i = \sum_{j=1}^n \alpha_{ij} v_j, \quad (1.12)$$

where α_{ij} is the attention weight from position i (query) to position j (key) and v_j is the value vector derived from x_j . Each representation y_i is therefore a context-dependent embedding of token i , formed as a weighted average of all tokens in the sequence.

From an interpretability perspective, the attention weight matrix $\mathbf{A}$ for self-attention is an $n \times n$ matrix whose entry α_{ij} indicates how much token i attends to token j . Visualizing $\mathbf{A}$ as a heatmap provides insights into which tokens influence each other. In neural machine translation, such attention maps often resemble word alignment matrices between source and target sequences [9, 85].

1.2.4 Multi-Head Attention and the Transformer Block

The transformer uses *multi-head* attention to allow the model to jointly attend to information from different representation subspaces at different positions [144]. Instead of

computing a single attention transformation, the model uses H parallel heads, each with its own projection matrices:

$$\begin{aligned} \mathbf{Q}^{(h)} &= \mathbf{XW}_Q^{(h)}, \quad \mathbf{K}^{(h)} = \mathbf{XW}_K^{(h)}, \quad \mathbf{V}^{(h)} = \mathbf{XW}_V^{(h)}, \\ h &\in \{1, \dots, H\}, \\ \text{head}_h &= \text{Attention}(\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)}). \end{aligned} \quad (1.13)$$

The outputs of all heads are concatenated and linearly transformed:

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\text{head}_1, \dots, \text{head}_H) \mathbf{W}_O, \quad (1.14)$$

with $\mathbf{W}_O \in \mathbb{R}^{Hd_v \times d_{\text{model}}}$.

A standard transformer layer includes multi-head self-attention followed by a position-wise feed-forward network (FFN), each sub-layer wrapped with residual connections and layer normalization (LN) [144]. Hence,

$$\begin{aligned} \mathbf{Z} &= \text{LN}(\mathbf{X} + \text{MultiHead}(\mathbf{X})), \\ \mathbf{Y} &= \text{LN}(\mathbf{Z} + \text{FFN}(\mathbf{Z})). \end{aligned} \quad (1.15)$$

1.2.5 Interpretability of Attention and Transformers

Empirical analyses show that different heads in large transformer models often specialize in capturing distinct linguistic or structural patterns, such as syntactic dependency relations, coreference links, or positional patterns [27, 146]. This specialization further motivates the inspection of individual attention heads for interpretability.

1.2.5.1 Direct Inspection of Attention Weights

One of the key interpretability advantages of attention mechanisms is that they explicitly compute weights α_{ij} that indicate how strongly each value v_j contributes to the representation of query q_i . In sequence-to-sequence models with attention [9, 85], these weights can be visualized as alignment matrices between source and target tokens, often revealing intuitive alignment structures that correspond to human-understandable translations.

In self-attention, the matrix $A \in \mathbb{R}^{n \times n}$ can be plotted as a heatmap. Such visualizations frequently show interpretable patterns, e.g., heads focusing on the previous or next token, or on syntactic heads and dependents [27, 146]. The fact that each row of A is a normalized distribution means that attention weights can be compared both within and across positions, and summary statistics such as *attention entropy* can be used to quantify how focused or diffuse the attention is. [95]

1.2.5.2 Limitations: Is Attention an Explanation?

Despite these appealing properties, there is a significant dissonance in the literature in the question if attention weights provide faithful explanations of model predictions, as already mentioned above. Jain and Wallace [61] demonstrate that it is often possible to substantially alter attention weights without materially changing the model’s output, suggesting that attention is not uniquely tied to the underlying causal mechanisms of the model. Similarly, Serrano and Smith [130] show that masking out tokens with high attention weights does not always cause large changes in model predictions, questioning the causal relevance of attention weights as explanations.

Wiegreffe and Pinter [150] argue that attention can still be useful for explanation under certain perspectives and evaluation criteria, but they also emphasize that attention-based explanations must be carefully validated, e.g., through *faithfulness* tests questioning how well attention-based rationales track actual changes in model behavior, or *plausibility* assessments questioning if they align with human judgments.

These results highlight that while attention weights are easily accessible and often intuitively meaningful, they should not be treated as definitive explanations without further analysis. Attention primarily exposes a *representation-level* mechanism, namely how context is aggregated, but this mechanism is embedded in a larger computation that includes non-linear transformations and multiple layers.

1.2.5.3 Aggregated and Layerwise Analyses

Transformers stack many layers of multi-head attention, so the effect of attention is compositional: information may flow across tokens through multiple cascaded attention steps. To better capture such effects, Abnar and Zuidema [1] propose methods such as *attention rollout*, which recursively multiply attention matrices across layers to estimate how much each input token ultimately influences each output token. This produces a more global notion of attention-based influence that is often more aligned with intuitive explanations than per-layer, per-head attention maps.

Other work investigates identifiability and redundancy in attention patterns, showing that similar outputs can arise from different attention configurations, and that some heads can be removed with little impact on performance [22, 27]. Such findings again emphasize that individual attention weights should be interpreted in context: certain heads may be more critical than others, and the overall behavior depends on how many heads and layers interact.

1.2.6 Attention as a Tool for Model Inspection

Despite these caveats, attention remains a powerful and practical tool for qualitative model inspection and hypothesis generation. Attention visualizations have been used to

- identify which tokens the model focuses on for specific predictions,
- detect spurious correlations, e.g., attention consistently focusing on unrelated artifacts,
- analyze cross-lingual alignments and structural patterns [9, 27, 146], and
- compare behavior across models or training regimes.

In many applications, attention-based analyses are complemented with gradient-based methods, feature ablations, and probing classifiers to obtain a more comprehensive picture of model behavior.

Taken together, the attention mechanism in transformers provides both a powerful representational tool and a rich object of study for interpretability. While attention weights alone do not fully explain a model’s decisions, their explicit, probabilistic nature and their geometric interpretation as similarity-based weighting make them an important and informative lens on how transformer models process and combine information.

1.3 Incorporating Knowledge into Data-Driven Models

Expert knowledge can be incorporated into data-driven models in many fashions. In literature, this concept is referred to as *Hybrid ML* [153], *Informed ML* [125] or *Theory-Guided Data Science* [69]. In classical purely data-driven approaches, the model learns patterns solely from the data having to explore the hypothesis space fairly “unguided”. This can lead to predictions that are physically inconsistent, implausible, or violate already known domain constraints that were not covered in the training data sufficiently. Moreover, purely data-driven models often require large amounts of training data to learn complex relationships, which can be a limiting factor in many real-world applications where data is scarce or expensive to obtain. This motivates the integration of knowledge into the learning process and/or model architecture.

But what actually is “knowledge” in particular? According to [125], knowledge can be categorized into three main types:

Scientific knowledge is typically formalized and validated by scientific experiments from established fields. Its most common representation in ML modeling is in the form of algebraic equations, differential equations, or simulation results.

World knowledge subsumes more general information that is usually common human knowledge like objects or subjects are perceived in the real world. Moreover, this category includes for instance language syntax and grammar. In ML research, this type of knowledge is often represented in the form of spatial invariances, logic rules, or knowledge graphs.

Expert knowledge is specific to a certain group of domain experts. It is often hard to formalize and more like intuitive understanding of processes, relationships, or patterns in the data. In ML modeling, expert knowledge is often represented in the form of probabilistic relations or human feedback.

It was shown by many branches of research that incorporating various types of knowledge into data-driven ML models enhances their performance substantially. [30, 69, 117, 120, 125, 131, 152] Generally, the knowledge can enter ML models at different stages of the modeling process:

Data and feature level With domain understanding of the process to be modeled, data can be preprocessed more effectively, e.g., by selecting relevant features, engineering new features based on known relationships, or augmenting data in a way that reflects domain-specific invariances. Furthermore, known relationships like thresholds and monotonicities may be encoded here.

Model and learning level At this level, knowledge can be integrated directly via the modeling approach itself. For instance, hybrid or “gray-box” models combine mechanistic or rule-based models with data-driven components to capture unknown or residual effects, also known as residual modeling [105]. Other approaches embed known constraints, laws, or structural relationships directly into the loss function or architecture, e.g., conservation laws or safety constraints [69, 117]. Moreover, prior distributions and regularization techniques can be used to mimic expert beliefs about plausible solutions.

Output and decision level At this stage, the output of the ML model can be post-processed or adjusted based on domain knowledge. For instance, predictions can be calibrated using known distributions, or filtered to enforce consistency with expert rules or safety requirements. Additionally, human feedback approaches may refine model outputs iteratively.

In this dissertation, the focus lies on incorporating knowledge at the data and feature level as well as at the model and learning level since phytopathological processes are well described in literature and expert knowledge is available that helps to unravel potentially “hidden” relationships and patterns in the data.

A nice side effect of incorporating knowledge into ML models is that resulting models often are more understandable and interpretable. By embedding known relationships and constraints, the model’s behavior can be more easily related to established domain concepts, making it easier for experts to trust and validate the model’s predictions. This interpretability is particularly important in fields where decisions have significant consequences, such as healthcare, finance, and environmental management. Moreover, the technology transfer from research into the real-world deployment becomes smoother when users can comprehend and verify the model’s reasoning. This low-threshold adoption is especially relevant for fields like agriculture, where end-users may not have deep technical expertise but possess substantial domain knowledge.

Being an ancient craft, agriculture is a well-researched field with versatile studies, well-described biological and phytopathological processes, and expert knowledge from practitioners. This rich background provides a solid foundation for integrating expert knowledge into ML models to improve their relevance and applicability in agricultural contexts. Thus, this dissertation should show how expert knowledge and data-driven modeling can be combined to solve key precision agriculture use cases with informed ML methods.

1.4 Goal and Structure

Now that the necessary background of this dissertation has been outlined, the goal and structure are presented in the following. As the subtitle of the thesis already suggests, the combination of knowledge and data-driven modeling is demonstrated through the analysis of plant disease spread in agricultural fields. The goal is to have an end-to-end workflow from georeferenced UAV and weather sensor data to an epidemiological analysis of diseases in the field. All model approaches should be formulated in a generalist manner so that they can be applied to different crops, diseases, fields, and climatic conditions. To demonstrate the principle, however, the focus is on the *Cercospora* leaf spot (CLS) disease on sugar beet [148]. The end-to-end workflow consists of three main building blocks summarized in Figure 1.1.

Plant Cataloging (Chapter 2) In this initial step, a semi-automated computer vision (CV) workflow to extract plant positions from large-scale UAV data is established in order to gather image time series of individual plants. This prerequisite step enables a massive RS data basis to be used for data-driven predictive modeling for precision agriculture use cases. In the scope of this thesis, the focus is on individualized images of sugar beet plants over the complete harvest cycles that are used for the following steps. The chapter contains content from the publication [A].

Disease Severity Prediction (Chapter 3)

In this building block, the gathered image dataset is used for automated image-based disease severity prediction. Since human-annotated disease severity (DS) is an expensive and time-consuming process, this step incorporates environmental data. The environmental data is used to calculate parameters that are proven to accurately reflect the plant development stage. During model training, those parameters are used as pretraining labels to reduce the need of large scale annotations. As a basis of the model, a vision transformer (ViT) is used to benefit from advantages of the attention mechanism. This chapter contains content from the publication [E].

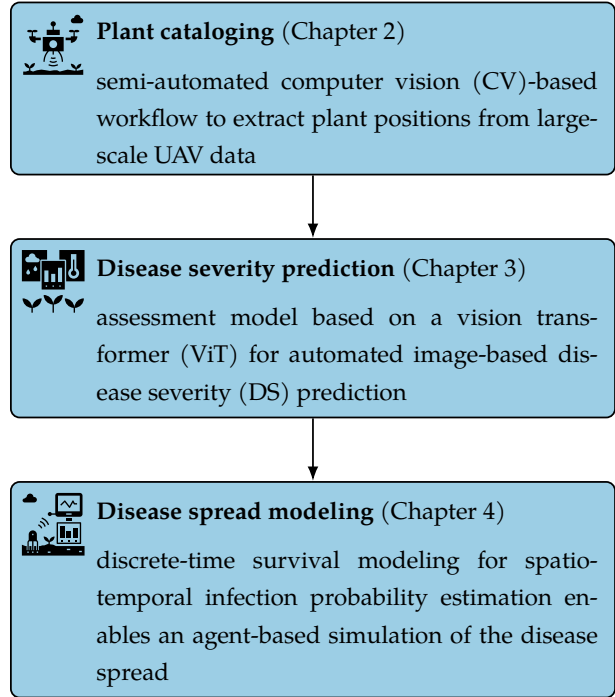


Figure 1.1: Main building blocks of this dissertation. The disease spread analysis consists of these three research parts which are built upon each other.

Disease Spread Modeling (Chapter 4)

Having large-scale spatio-temporal DS predictions, the spread dynamics can be modeled. Phytopathological knowledge—like relation between incubation time and environmental parameters—is used to retrieve the infection day of individual plants which is not visible per se. Thereafter, environmental data is used to formulate meaningful parameters. Combined with spatio-temporal knowledge, a tailored model is developed using again the power of attention to learn the disease spread dynamics. The resulting model can be used in an agent-based simulation to predict the disease spread in the field. This chapter contains content from the publication [B].

In the end, Chapter 5 concludes the dissertation with a summary of the main findings, a discussion of limitations, and an outlook on future research directions.

1.4.1 Cercospora Leaf Spot in Sugar Beet

The spread of the CLS disease in sugar beet is used to demonstrate the workflow. This is not a random choice since CLS critically affects sugar beet yield. It is caused by the pathogen *Cercospora beticola* Sacc. which is the most destructive foliar pathogen in sugar beet all over the world and reduces quality and yield of sugar beet farming [148]. Symptoms of CLS appear as numerous small, round, gray spots with a red or brown border on leaves. As the infestation increases, the leaves become necrotic and dry up. When a large

part of the leaf area is lost, the plant often tries to recover by generating new leaf shoots at the cost of its stored sugar. However, if the conditions are favorable for the fungus and the attack is very severe, the plants die. [119, 148] Figure 1.2 shows a leaf of a CLS-infested sugar beet. Saccardo first characterized the fungus in 1881 [127]. Since then, many studies have been conducted to understand disease cycle, epidemiology, and control measures of CLS [12–14, 119, 132, 155, 157]. Research into this disease is particularly important because the sugar beet itself is an important crop in today’s global agriculture. In 2024 alone, global sugar beet production was estimated at around 294 Mt. The three most important producers in the world are Russia (46.7 Mt), Germany (36.7 Mt), and France (32.5 Mt). Moreover, it is—by production quantity—the most profitable crop in Germany. By sugar beet yield, Germany is the second most fertile country (83.9 t/ha) after Chile (94.4 t/ha) [37].

All in all, CLS is a major threat to agriculture and food security and, thus, is worth studying in detail. In the next section, it is explained how the severity of CLS is estimated.

1.4.2 Disease Severity Estimation

In precision agriculture, the use of UAVs equipped with multispectral cameras for monitoring agricultural fields is well-established for various tasks regarding plant phenotyping and health status. The usage of multispectral instead of pure optical cameras reveals important insights into those spectral regions where differences in the reflectance between healthy and infested plants are more remarkable than in the visible light spectrum. [90] Especially in phenotyping for breeding, one of the main advantages of UAV imagery is the mapping flexibility in comparison to satellite imagery, automation and homogenization of laborious and time-consuming visual scoring activities, which usually require large numbers of hours of specialized human labor to score large fields. In agriculture, the term “visual scoring” is commonly used for a field assessment, such as phenotyping canopy structure or the quantification of disease intensity, specifically, disease severity (DS) [16]. In the field of (supervised) ML, a related procedure, called “annotation”, is used. Annotation consists of labeling data elements in order to add semantic information or metadata. In essence, although the terms differ in their apparent applications, they represent an equivalent concept, and in this context, let “annotation” be a synonym for “scoring”. Theoretically, when large field experiments are conducted, such data is immediately available. However, a lot of human-powered effort is needed to gain information out of large imagery. This is where ML comes into play.



Figure 1.2: Symptoms of CLS on a sugar beet leaf. Taken from [73].

Generally, the assessment of DS is a fundamental task in plant phenotyping and plant disease management [104]. It is essential for monitoring disease progression and making informed decisions regarding crop management. Traditional methods for assessing DS often rely on visual scoring by human experts, which can be subjectively biased, time-consuming, and labor-intensive [15, 104]. Therefore, there is a growing interest in developing automated and objective methods for DS assessment using advanced technologies such as RS interconnected with modern ML techniques [133].

In this work, the goal behind the use case is to determine the DS in CLS-infected sugar beet fields. Usually, the observation and assessment of plants is done by visual scoring. Visual field scoring of DS, an activity that requires a lot of time and well-trained personnel [104], is a main bottleneck in the control of CLS. Therefore, it is desirable to have an automated DS estimation model enabling large-scale and precise monitoring of the disease stage in the field.

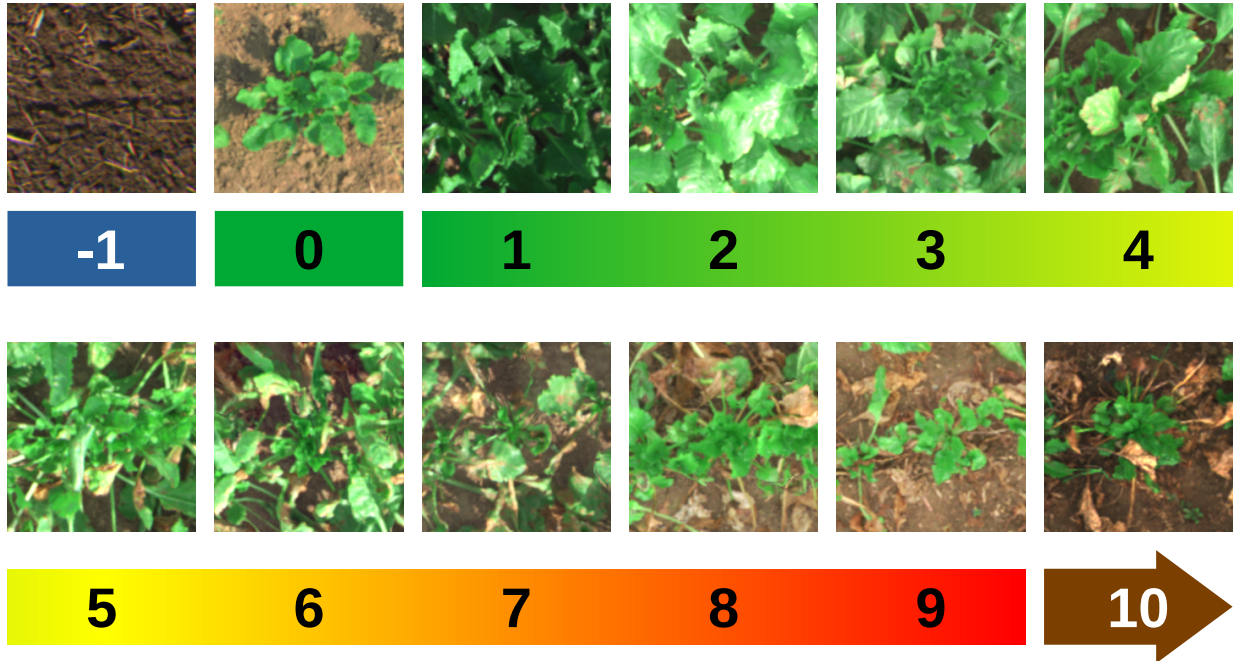


Figure 1.3: Used disease severity (DS) scale for the prediction model with example images. The scale is based on the usual CLS rating scale complemented with the definitions for -1, 0 and 10.

Considering heterogeneous disease distribution within sugar beet fields, a detailed and geo-referenced assessment of DS might lead to precise protection measures within the canopy. Geo-referenced and plant-based determination of DS is therefore essential. A naive approach for the prediction of DS with single plant images would be to model a classification problem like in [147]. Despite being a valid approach at first sight, certain phenotypical knowledge enables a more intelligent modeling. Thus Chapter 3 will elaborate a basic paradigm to solve the DS estimation problem and a proposed deep learning model based on it.

First, a DS annotation scale that serves as a guideline for all human expert annotations and finally as “unit” of the model input has to be defined. The DS assessment in this work will use the rating scale developed in [G] with an extension for non-infested and newly sprouted plants. Figure 1.3 shows the numerical scale with exemplary plant images. The scale from 1 to 9 belongs to definitions of the KWS scale. It was established and named after the German company *Kleinwanzlebener Saatzucht*¹. [75] The KWS scale is a severity diagram that ranges from 1 to 9. A rating of 1 indicates the complete absence of symptoms, while a rating of 3 indicates the presence of leaf spots on older leaves. A rating of 5 signifies the merging of leaf spots, resulting in the formation of necrotic areas. A rating of 7 is assigned when the disease advances from the oldest leaves to the inner leaves, leading to their death. Finally, a rating of 9 is given when the foliage experiences complete death [75]. In order to complement the scale, a “0” for non-infested sugar beets before canopy closure, and a “10” for newly sprouted plants as in [G] is added. In order to apply the model also on regions, where only soil is visible, a “-1” as “no plant” or “soil only” label is further added by still maintaining the continuous fashion of the severity scale. This should also increase the model’s focus on the plant because it learns examples where no plant is visible at all.

In field experiments, one often faces data or annotations that require substantial effort to acquire. Nevertheless, some data can be acquired rather automatically or with low human effort. This work distinguishes between “cheap” and “expensive” labels. The DS acquisition is rather expensive, while, for instance, weather data acquired with automatic sensors or public weather stations is, typically, relatively cheap. Additionally, the development and epidemiology of the pathogen and disease CLS is highly influenced by specific environmental conditions. [156] Thus, an intelligent approach will make use of the cheap data in order to increase efficiency on expensive data. In the next section, the data basis that is used in the dissertation is described in detail.

1.5 Data Basis

Particularly in precision agriculture, multimodal data are used, i.e., data from diverse sources in diverse data structures and representations. This work mainly focuses on two modes of data: (1) multispectral reflectance imagery from UAV systems and (2) environmental data from IoT weather sensors, weather stations represented as time series data. Since a fungal disease driven by spore infections is investigated, spore traps are installed in the fields as well. Their data is again represented by time series. Moreover, human annotated DS scores on individual plants are used as ground truth data for model training and evaluation. The data taking effort can be very different which is

¹now officially KWS SAAT SE & Co. KGaA

why this work uses the terms of *cheap* and *expensive* data. With cheap data, types of data are subsumed that are gathered rather time- and cost-efficient and involve low human intervention effort during the data taking process. This includes data from IoT weather sensors and weather stations but also the UAV imagery. Contrary to this, expensive data means data connected with high human effort or expensive data taking and/or processing, for instance, human DS assessment on individual plants in the field or by images or spore concentration measurements. Such data is often scarce and limited in quantity.

The dataset used in this work comprises multispectral images expressed in reflectance of single sugar beet plants recorded by UAV systems on 6 different locations near Göttingen, Germany (51°33'N 9°53'E), across multiple sugar beet field experiments focused on CLS disease. The used UAV for the aerial data collection is the quadcopter *DJI Matrice 210*². The position of the UAV during the flight was determined by GPS and global navigation satellite system (GLONASS) and corrected by the *D-RTK 2 Mobile Station*². The camera mounted is a *MicaSense Altum*³ multispectral camera with 6 bands: blue (32 nm bandwidth), green (27 nm bandwidth), red (14 nm bandwidth), red edge (REDGE) (12 nm bandwidth), NIR (57 nm bandwidth), and LWIR with a *Downwelling Light Sensor* (DLS2). For early experiments, a *MicaSense RedEdge-M*³ multispectral camera has been used. Side-overlap and forward-overlap were set to 75 %. In addition, ground control points (GCPs) were set up at the corners and in the middle of the test arrangement for the multitemporal monitoring in the sense of georeferencing and correcting the collected data. *Agisoft Metashape Professional*⁴ software has been used to create orthomosaic images. Due to a large amount of investigated sugar beet varieties, resolutions—expressed by ground sampling distance (GSD)—, locations, and time points, the dataset is very diverse. In total, it covers 4 harvesting periods from 2019 to 2022 and comprises 17 different experiments or flight missions. Table 1.1 gives an overview of the most important information of the dataset. Additionally, Table 1.2 shows the spectral bandwidths of the two camera sensor systems used in this work.

The environmental data includes several weather and microclimate measurands coming from a local weather station and multiple *CLIMAVI* IoT sensors⁵ [2]. They are placed inside the fields at a height of 50 cm measuring the microclimate. As an addition for observing the CLS disease spread, spore concentration measurements were recorded using spore traps installed in the fields. Moreover, local daily weather data from public stations is used as further environmental parameters. A detailed overview of the recorded measurands and their measuring frequency is listed in Table 1.1. Furthermore,

²SZ DJI Technology Co., Ltd., Shenzhen, China

³MicaSense, Inc., USA

⁴Version 1.6.3.10732

⁵Agvolution GmbH, Germany

Measurand	Unit	Source	Multiplicity	Frequency
Temperature	°C	IoT sensors	multiple	1/h
Relative humidity	%	IoT sensors	multiple	1/h
Wind direction	°	weather station	single	1/h
Wind speed	m/s	weather station	single	1/h
Precipitation	mm	weather station	single	1/h
Global radiation	W/m ²	weather station	single	1/h
Leaf wetness	%	weather station	single	1/h
Spore concentration	ng	spore traps	multiple	1/d

Table 1.1: Overview of recorded environmental data used for the sugar beet field experiments. Sensors with “multiple” multiplicity are installed at multiple locations in the fields, while sensors with “single” multiplicity are only installed at one location somewhere outside the field.

Band Name			Camera	
Long		Short	Altum	RedEdge-M
blue		B	459 – 491 nm	465 – 485 nm
green		G	548 – 572 nm	550 – 570 nm
red		R	661 – 675 nm	663 – 673 nm
red edge		REDGE	711 – 723 nm	712 – 722 nm
near infrared		NIR	814 – 870 nm	820 – 860 nm
long-wave (thermal) infrared		LWIR	8 – 14 μ m	—

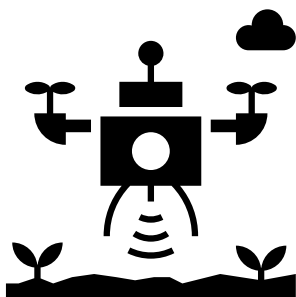
Table 1.2: Spectral ranges of the two used camera sensor systems. The thermal band is not used in this work.

human interventions like fungicide applications and manual application of CLS by inoculation were documented. In total, the dataset contains 4 harvesting periods from 2019 to 2022 and comprises 17 different experiments or flight missions. Table 1.3 outlines all important information of the dataset and how the different experiments are used for model training, validation, and testing.

The following steps of this thesis manipulate, enrich and process this data for their respective use cases or considerations. Inside the chapters, further use-case-specific data processing may be described in detail.

ID	Sowing Date	Location	Disease Severity Estimation								Spread Modeling		
			Sensor	GSD mm/px	Varie- ties	Treat- ments	Total Area m ²	Flights	Images	Split	Plants	Annota- tions	Split
D01a	2019-04-09	Holtensen I	RedEdge-M	3.5	2	6	580	23	13 428	train/val			
D01b	2019-04-09	Holtensen I	RedEdge-M	15	2	6	580	21	18 157	train/val			
D02	2020-04-06	Weende	Altum	3	2	6	580	25	37 905	train/val			
D03	2020-04-06	Weende	Altum	4	1	3	1970	25	240 120	train/val	37 020	24	train/val
D04	2021-04-01	Sieboldshausen	Altum	5.1	51	102	3500	7	39 527	train/val			
D05	2021-04-23	Holtensen I	Altum	3	5	15	960	27	112 655	train/val			
D06	2021-04-23	Holtensen I	Altum	4	1	4	1970	23	212 559	train/val	13 330	16	train/val
D07a	2021-04-23	Holtensen I	Altum	3	1	2	600	22	35 878	train/val	3270	11	train/val
D07b	2021-04-23	Holtensen I	Altum	5	1	2	600	22	45 701	test			
D08a	2021-04-30	Dramfeld	Altum	4	1	2	600	22	33 484	train/val	3053	11	test
D08b	2021-04-30	Dramfeld	Altum	5	1	2	600	22	30 530	train/val			
D09a	2022-03-31	Reinshof	Altum	9	1	2	24 000	11	11 406	train/val			
D09b	2022-03-31	Reinshof	Altum	18.5	1	2	24 000	8	7065	train/val			
D10	2022-03-31	Holtensen II	Altum	3.4	1	2	4350	21	59 639	train/val			
D11	2022-04-19	Weende	Altum	4	1	4	2220	12	203 139	train/val	12 348	16	train/val
D12	2022-04-20	Reinshof	Altum	5	4	1	975	11	21 301	train/val			
D13	2022-04-20	Weende	Altum	5.1	4	1	982	12	15 604	train/val			
Total									1 092 397 45 701	train/val test	65 968 3053	67 11	train/val test

Table 1.3: Overview of datasets used for disease severity estimation and spread modeling. The disease severity estimation datasets contain human annotated disease severity scores on individual plant images extracted with the cataloging workflow.



2

Plant Cataloging

The content of this chapter is based on the publication [A]. My work in this publication contained the conceptualization of the plant cataloging workflow, the design and implementation of the software, and the writing of the manuscript. The publication was a collaboration with colleagues from the Institute of Sugar Beet Research (IfZ) in Göttingen, and the Institute of Geodesy and Geoinformation (IGG) of the University of Bonn, where the data was acquired. Furthermore, the workflow is now used by the IfZ for cataloging plants and the retrieval for their positions in upcoming sugar beet field experiments.

2.1 Introduction

As the initial building block of an end-to-end workflow for plant disease spread prediction, the plants themselves have to be identified and individualized in the unmanned aerial vehicle (UAV) imagery. Typically, the observed field area is large and recorded on several time points during the full harvest cycle, so that manual labeling of the plants is tedious or even infeasible. Thus, a framework for semi-automated recognition of plant positions is presented in this chapter. Semi-automated denotes that the recognition is based on predefined parameters to be set by the user. Those comprise easily accessible knowledge about the underlying field structure like estimated average seeding distance

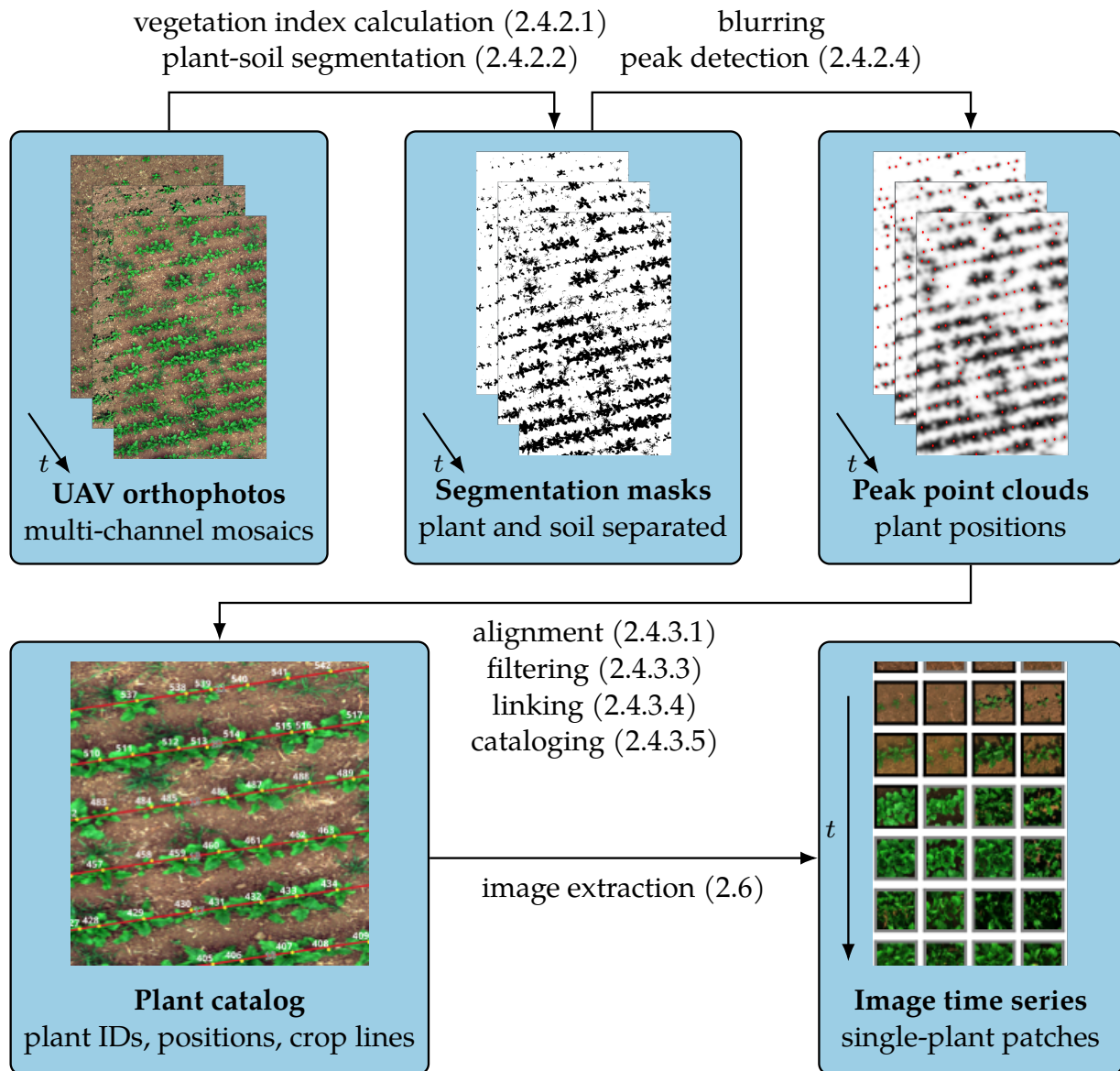


Figure 2.1: Overview of plant cataloging and image extraction workflow. The multi-channel unmanned aerial vehicle (UAV) images with different acquisition dates are processed for plant-soil segmentation. Afterward, adaptive Gaussian blur filtering helps to locate the plant positions more precisely via peak finding. Subsequently, the peaks are grouped by time and further manipulation like seeding line detection leads to the plant catalog. Once this catalog is available, one can extract image time series data of single plants. The numbers in parentheses refer to the respective sections in this chapter.

(intra-row spacing) of the crop and the estimated crop line distance (inter-row spacing). Additionally, a few tuning parameters for the image processing and filtering have to be adjusted. Once this is done, the workflow detects the plant positions and retrieves them in the complete image series over the acquisition dates so that the images can be registered on each other. As a result, the plants can be individualized and allocated in a spatial and temporal sense, referred to as *cataloging* the plants. It enables the generation of large-scale image datasets that can be further used for machine learning tasks.

2.2 Background

To observe individual plants, the exact positions of the plants in the image are required. At the moment, it is possible in two arduous and time-consuming ways: (1) by georeferencing the plants with sub-centimeter accuracy directly on the field by using a GPS receiver with real-time kinematic (RTK) positioning or (2) after image acquisition and orthorectification by manually annotating the plants in a geographic information system (GIS). When taking time series images, the perfect alignment of these images requires the use of well-known and measured points, called ground control points (GCPs) [107]. If the GCPs are correctly georeferenced, they allow allocation of the images not only in a local but also in a global coordinate system, such as the world geodetic system 1984 (WGS84). This enables pixel-precise transferability between images from different locations and time points. The localization of a plant is, thus, only necessary at one point in time. The determined absolute coordinate remains identical at every point in time, since the plant does not move in space. Hence, a simple extraction of the plant at different points in time is possible. Overlapping of neighboring plants does not affect localization in this approach if the position is calculated in an early growth stage. If no georeferencing of the data is available, e.g. because no access to measuring devices or GCPs is possible, another possibility is to match the images using plant position detection based on single images. In a further step, these detection points can be registered with each other. However, dealing with overlaps of plants in later developmental stages is challenging in this approach.

Once the different time points are aligned, it is feasible to extract time series for each individual plant. Time series allow the analysis of the plant growth and especially the assessment of the plant traits and their temporal development. Based on time series, it is possible to identify the individual development of each plant. This spatio-temporal information enables optimization of crop management. Another aspect is the detection of stress factors and disease development, and how it influences the growth of the affected plant over time.

This chapter introduces a complete workflow for plant identification and individualization—further also referred to as “cataloging”—of single plants. Figure 2.1 summarizes the proposed workflow with its main steps. Afterward, the workflow is evaluated on ground truth data.

2.3 Data Description

As introduced in Section 1.5, the CLS dataset on sugar beet is used, specifically the set D03. QGIS¹ [114] was used to delimit and select the area to be analyzed, resulting in the dataset being split into 19 subplots with an equal area of 172 m² each. The plot trials were systematically conducted with the different treatment strategies: (1) control with fungicide, (2) inoculated with fungicide, and (3) inoculated without fungicide. Plots of inoculated treatments were infected manually with CLS-diseased sugar beet air-dried leaf material. UAV flights were conducted on a weekly basis at 24 dates during the complete growing season from May 7th to October 12th, 2020. As the crop, one variety of sugar beet plants that is susceptible to CLS was selected—the *Aluco*² variety. The sugar beet was sown on April 6th, 2020, with an inter-row distance of 48 cm, an intra-row distance of 18 cm, and an expected seeding rate of approximately 110 000 plants per hectare. Figure 2.2 shows the location of the plots in the trial field. For the quality evaluation of the workflow, some fields are annotated by human experts, preferably in the earlier stages where the plants are locally distinguishable, by marking the central position of each visible plant.

In the original publication [A], another dataset is used that monitors the development and harvest time of cauliflower. Such data can be used to extract phenotypic traits such as diameter or height of cauliflower plants or their head [32, 49]. Additionally, plant disease research in general is an excellent use case for this workflow since it requires having individual plant information throughout many growth stages [17, 18, 88].

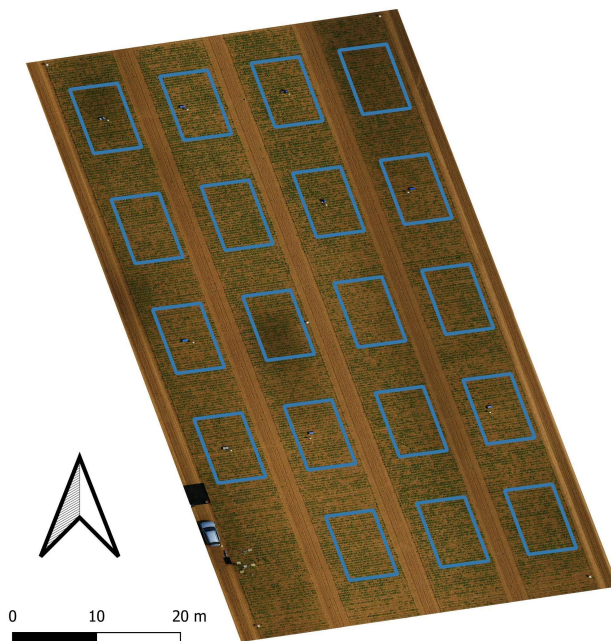


Figure 2.2: Overview of Cercospora leaf spot (CLS) experimental field 2020 in Göttingen, Germany (from dataset D03). The 19 marked areas correspond to the plots under analysis and undergo different treatments like disease inoculation and fungicide application. The measurements were conducted throughout the full growing season.

¹Version 3.16

²SESVanderHave

Thus, a spatio-temporal individualization of the investigated plants is crucial. Especially for large-scale investigations, an approach that is as automated as possible is desirable.

Some challenges for both datasets are, for example, the varying exposure due to the different data acquisition times during the whole growing season and the related different weather conditions. Occurring weeds can have positive as well as negative effects. On the one hand, it simplifies tasks such as registration, since weeds are stationary and thus serve as markers. On the other hand, the growth of weeds affects tasks such as plant detection, since a distinction between weeds and crops must be made beforehand. Another challenge occurs when the plants are so large that the canopy of the plants is closed. This complicates the separation of different plant instances and the search for distinctive points in the field, which is often made possible by the contrast between soil and plants.

2.4 Plant Extraction Workflow

This section will describe the full plant extraction workflow in detail. For each workflow step, plots on different excerpts from the sugar beet dataset visualize the results. Figure 2.1 can be read as a flowchart for the workflow, where the captions on the arrows list the steps top to bottom.

2.4.1 Preprocessing

Some datasets may require preprocessing steps before the workflow can be applied effectively. Firstly, one obviously needs georeferenced data in the form of so-called orthophotos. These are flattened, distortion-corrected images of the surface generated from the UAV images by using the software *Agisoft Metashape Professional*, as mentioned in Section 1.5. Secondly, it is beneficial—especially for the seeding line detection that will be discussed later—to crop the orthophotos to the region to be processed by the workflow. Any foreign objects or tramlines should preferably be removed. Foreign objects could have similar color properties as the plants and may lead to false recognitions. Furthermore, tramlines may cause irregular seeding line distances, which could lead to inaccurate filtering. Additionally, cropping accelerates the computation since less image data has to be processed by the workflow steps.

2.4.2 Plant Position Detection

As the first step of plant cataloging, the plant positions are determined for each acquired image, i.e. for each plot at each acquisition date. For this, a segmentation between soil

and plants is performed. Common approaches use vegetation indices (VIs) as a preprocessing step for the image information [50]. There are pure red, green, and blue channel (RGB)-based VIs as well as VIs including multi- or hyperspectral information. Due to the multispectral image acquisitions in the sugar beet dataset, the contrast between plants and soil could be increased compared to RGB-only images. However, the experiments showed that including multispectral information did not lead to better results than using pure RGB information. This is evidence that the workflow performs well even for use cases where beyond-RGB imaging is not feasible or available.

2.4.2.1 Vegetation Index

Using spectral information to condense the information in a one-channel image is the main concept of VIs. Besides reducing the image dimension, the goal behind introducing the VI is to enhance the contrast between soil and plants compared with a standard RGB image, for instance. There are plenty of publications of different vegetation indices so far [59, 161]. However, they all combine different spectral channels by a certain calculation specification. In addition, some indices use (empirically motivated) parameters to further optimize them for different use cases. Since it should be demonstrated that the plant extraction method works for RGB-only information already, let us stick to RGB-based VIs. Two indices that turn out to work well in the experiments are the green leaf index (GLI) [83]

$$\text{GLI}_i = \frac{2G_i - R_i - B_i}{2G_i + R_i + B_i}, \quad (2.1)$$

as well as the normalized green/red difference index (NGRDI) [141]

$$\text{NGRDI}_i = \frac{G_i - R_i}{G_i + R_i}. \quad (2.2)$$

These indices map the information of red (R), green (G), and blue (B) channels of each image pixel i onto a single value in the interval $[-1, 1]$. If there is multispectral data available, for instance the near infrared (NIR) channel, nevertheless, one may use the optimized soil-adjusted vegetation index (OSAVI) [122] defined as

$$\text{OSAVI}_i = \frac{\text{NIR}_i - R_i}{\text{NIR}_i + R_i + Y}, \quad (2.3)$$

where $Y > 0$ is an empirical parameter. The authors of the original publication [122] found $Y = 0.16$ to be the optimal choice. Although it is not necessary to work with multispectral VIs in this stage, they can further discriminate plants from foreign objects in the UAV image data. Figure 2.3 shows the mentioned VIs applied on an example

image snippet taken from the sugar beet dataset. The remarkable points are that all three VIs can handle shadows fairly well, but the OSAVI is slightly better in ignoring foreign objects.

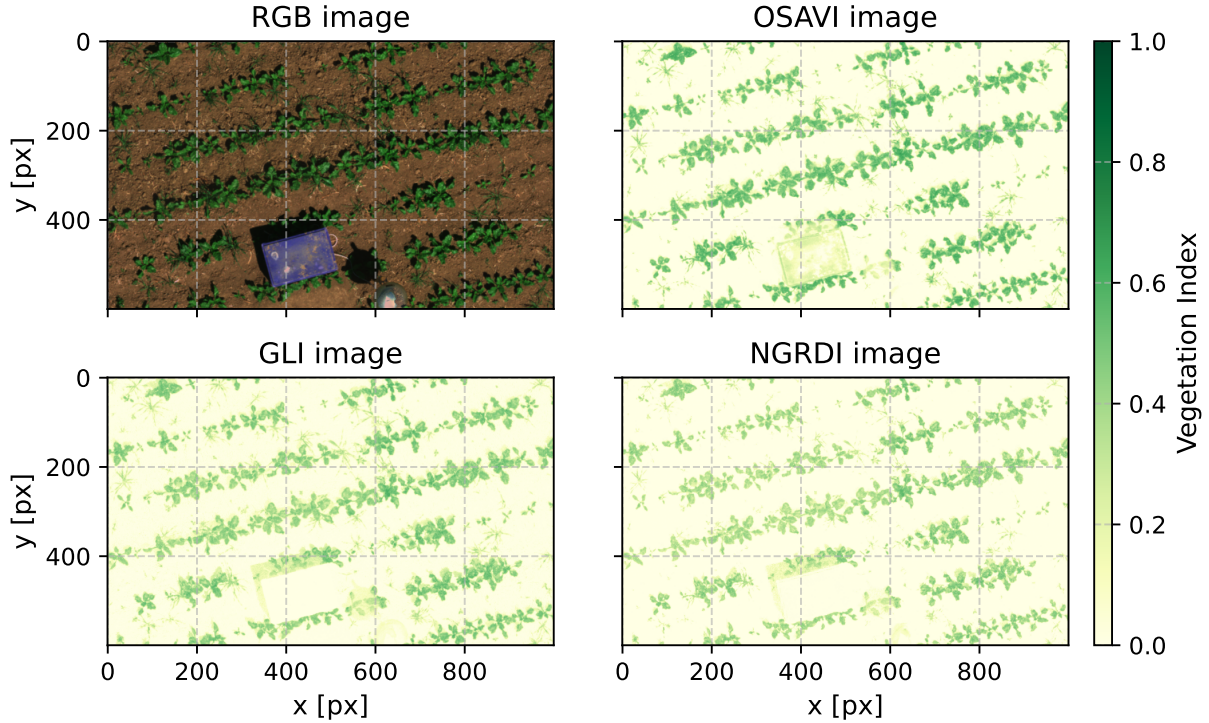


Figure 2.3: Different VIs for discrimination between plants and soil. The RGB image information (upper left) and three Vegetation Indices are shown. The OSAVI as a multispectral index (upper right) as well as the RGB-only indices GLI (lower left) and NGRDI (lower right) can be calculated for the multispectral sugar beet dataset. Using multispectral information, OSAVI is less susceptible to foreign objects in the field than RGB-only indices.

Note that other VIs that produce substantial contrast between plants and soil could also be used at this point. However, the further plant finding procedure restricts itself to the RGB-only VI images at this place. They are named as $\mathbf{V}(t)$ for each acquisition date $t \in \mathcal{T}$ in the set of given acquisition dates $\mathcal{T}$. The single pixel values are named by v_i .

2.4.2.2 Plant-soil Segmentation

In the next step, the soil has to be discriminated from the plant area. Thus, a segmentation is performed.

As Section 2.4.2.1 may already have suggested, the ratio between plant and soil coverage in the images is highly variable for an image series during a complete growing season. Therefore, it is important to keep the growing state—or the plant size—in mind for each image. Thus, a convenient measure is the cover ratio c defined by

$$c = \frac{1}{N_{\text{px}}} \sum_i^{N_{\text{px}}} \chi_{\{v_i \geq \vartheta\}}, \quad (2.4)$$

where v_i is the VI value for the i -th of N_{px} total pixels and ϑ is a predefined threshold value. $\chi_{\{\cdot\}}$ is the indicator function, which is 1 if the condition given in the index is met and 0 else. Descriptively, the cover ratio is just the percentage of pixels above threshold.

As threshold technique, Otsu's method is used [109]. The main idea is to find the threshold that minimizes the intra-class variance of two classes to be considered as foreground and background. For intermediately covered fields, this method leads to a reliable discrimination between soil and plants. However, it has weaknesses for the following extreme cases. For very sparsely covered fields ($c < 1\%$), Otsu's method usually results in a threshold that is approximately at the most frequent value—i.e. the mode—of the VI values. This is because the small amount of “plant pixels” simply disappears in the value distribution, which then has a single peak structure. In those images, the most frequent value is at soil level; thus, one would obtain much noise by using Otsu's threshold and would overestimate the cover ratio calculated by the resulting segmentation mask. This problem is bypassed by setting the threshold to the 99-percentile of the VI distribution, which obviously results in a modified cover ratio of exactly 1%. For cover ratios above circa 75%, single plants or seeding lines are not properly distinguishable anymore. Therefore, those stages are not used for segmentation.

Due to these limitations, an initial estimate of the cover ratio is needed in order to classify the images. For the two VIs considered in this work, good initial thresholds for both datasets were investigated. As a result, the initial thresholds are set to $\vartheta_{\text{GLI}} = 0.2$ and $\vartheta_{\text{NGRDI}} = 0$, respectively. For OSAVI, the initial threshold would be set to $\vartheta_{\text{OSAVI}} = 0.25$. The preliminary estimation for c decides which threshold method is used and the final cover ratio for each image is calculated subsequently. Figure 2.4 summarizes the findings by some exemplary field sectors during different growing stages.

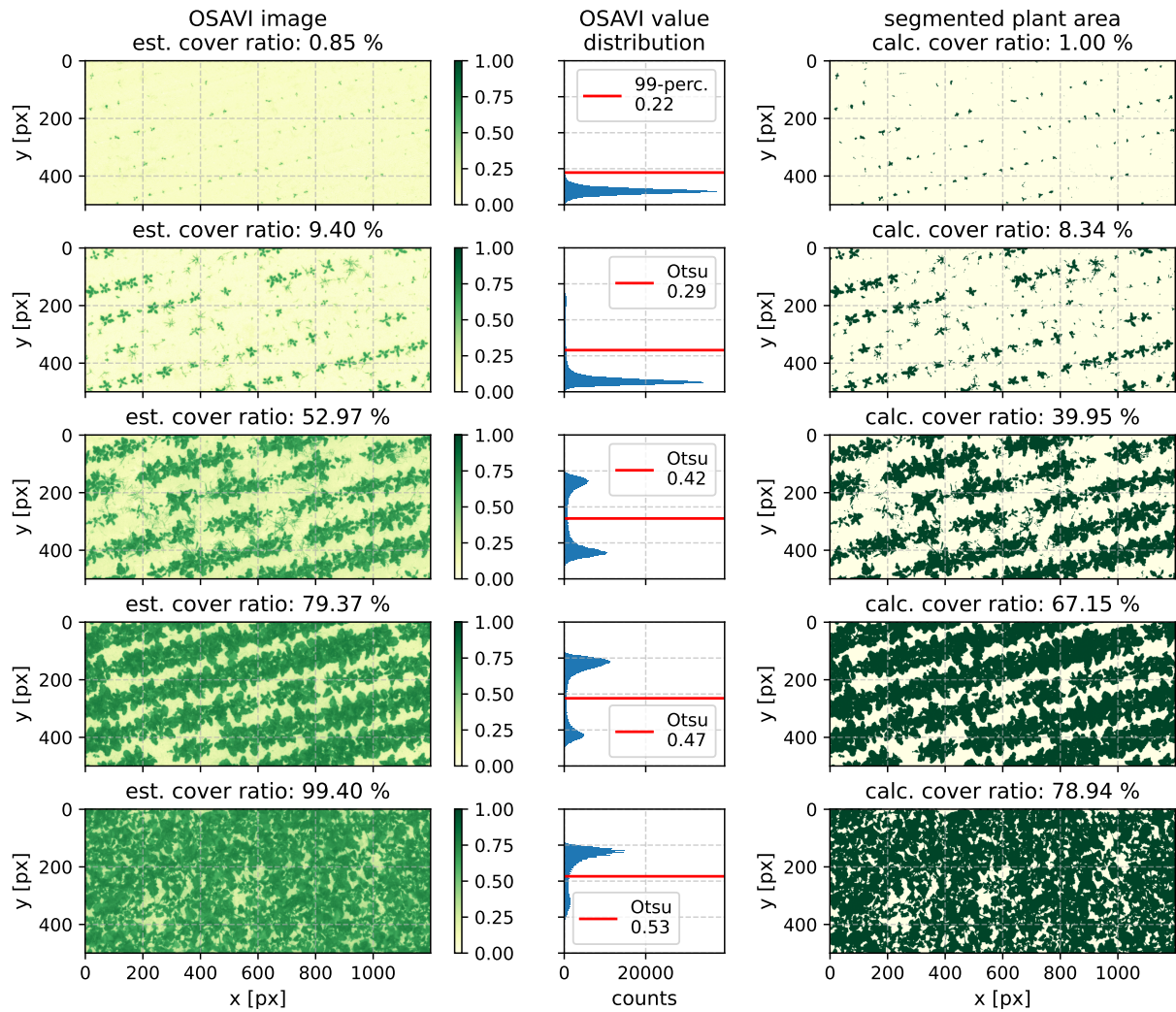


Figure 2.4: Plant-soil segmentation for different growing stages. The left plots show the OSAVI image. The middle histograms show the distributions of pixel values and the applied threshold for the segmentation—as seen in the right plots. In the top example, the estimated cover ratio is below 1 % so that the 99-percentile value is used as threshold. In cases depicted in the next three plots, Otsu’s method is applied. Until roughly 75 % cover ratio, single plants and seeding lines are visible. Images beyond that limit like in the bottom plot are not used for further analysis.

2.4.2.3 Growth Function

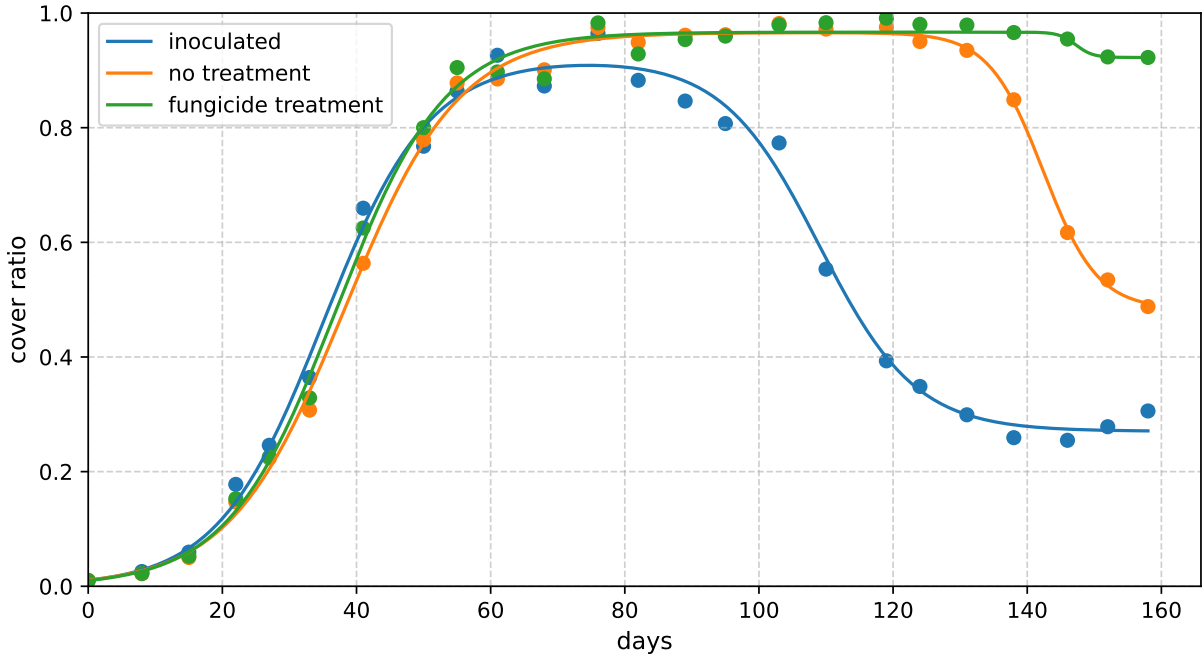


Figure 2.5: Growth function curve fits. The cover ratio estimates vs. days since first acquisition is fitted with the growth function (2.5). The shape of the growth function is different for the three categories. The blue curve shows an inoculated field. After the growing phase, the dying process due to the disease starts and saturates in the latest acquisitions. In case of the untreated field (yellow curve), the natural dying process is not saturated. In the fungicide treated field shown by the green curve, no dying phase is visible. Thus, the second dying term of the fit function is not present, in contrast to the first two cases.

If enough images during a complete growing season are acquired, a functional relation between the single estimated cover ratio can be estimated. Empirically and plotted against the acquisition date, the cover ratio follows a saturated exponential function until the plants are dying or suffering diseases. From then on, the cover ratio decreases exponentially. These phenomenological considerations motivate the approach for a *growth function* $c(t)$ defined as:

$$c(t) = \frac{g}{1 + \exp(-\lambda_g(t - t_g))} - \chi_{\{d>0\}} \frac{d}{1 + \exp(-\lambda_d(t - t_d))}, \quad (2.5)$$

where $\chi_{\{d>0\}}$ is the indicator function that is 1 for $d > 0$ and 0 else. Basically, two independent sigmoid functions approximate the growing and dying phase. The difference of them is the complete growing function. The growing (dying) slope constants λ_g and g (λ_d and d) as well as the corresponding time offsets t_g (t_d) are treated as optimization parameters. By separating cases for positive and non-positive d , one can give the optimization process the option to ignore the dying phase if it is not visible. For the time t , the days since the first acquisition are used. Figure 2.5 shows the growth function fit for 3 fields with different treatments.

2.4.2.4 Filtering and Peak Finding

As segmentation masks are calculated, one can find single plants by applying filtering and peak finding techniques. Recap that the segmentation masks (cf. Section 2.4.2.2) are represented as binary images

$$\mathbf{B}(t) = \left\{ \chi_{\{v_i \geq \vartheta(t)\}} \right\}_{i=1}^{N_{\text{px}}} \quad (2.6)$$

by applying Otsu's or 99-percentile threshold $\vartheta(t)$ for each VI image pixel v_i and acquisition date t . The resulting binary images $\mathbf{B}(t)$ can then be blurred with a Gaussian filter. Mathematically, the images $\mathbf{B}(t)$ are convolved with a 2D Gaussian kernel

$$\mathcal{G}(x, y, t) = \frac{1}{2\pi\sigma^2(t)} \exp\left(-\frac{x^2 + y^2}{2\sigma^2(t)}\right), \quad (2.7)$$

where $\sigma(t)$ is the bandwidth parameter. x and y represent the 2-dimensional image pixel dimensions. Note the time dependency here, since it makes sense to adapt the bandwidth with the size of the plants—i.e. given by the cover ratio estimate—and by introducing an interval of reasonable bandwidth for the given images $[\sigma_{\min}, \sigma_{\max}]$ in units of image pixels.

As a proxy, the given bandwidth boundaries should roughly represent the size of the plants in the images of the beginning growing stage and maximum growing stage, respectively. Finally, one gets the blurred images $\tilde{\mathbf{B}}(t)$ by convolution $(*)$ of the binary images $\mathbf{B}(t)$ with the Gaussian kernel matrix $\mathbf{G}(t)$, hence

$$\tilde{\mathbf{B}}(t) = \mathbf{B}(t) * \mathbf{G}(t). \quad (2.8)$$

Subsequently, a simple peak finder is applied to extract the plant centers. The blurring has the effect that the plants are visible as blurred quasi-circular objects. In this way, the peak finding algorithm detects the plant center rather than individual leaves. To further improve the results of this peak finding, a minimal peak distance and/or intensity can be set to avoid double detection of bigger plants or still visible weed.

Nevertheless, the outcome of this method is highly dependent on choosing the ranges for binary thresholds and Gaussian filter bandwidth correctly. In order to make this more user-friendly and adaptive to new datasets, it is beneficial to use metric units for the parameters and convert them into pixel units by the given geospatial information. Thus, the workflow user can refer to real-world plant sizes independently of image resolution. A good choice for $\sigma_{\min}$ and $\sigma_{\max}$ are found to be the radii of a seedling and a fully grown plant, respectively.

Moreover, this method is generally performing significantly better on images with lower plant cover ratios. For higher cover ratios, the plants are no longer spatially divided, which results in misdeterminations or multi-detections. As stated in Figure 2.4, single plants are hard to discriminate above circa 75 % cover ratio. Thus, this method is only performed on the lower cover ratio images. Figure 2.6 shows plant detection results for different exemplary growing stages. It happens that also unwanted weed is recognized by this method or not every single plant is detected. However, the steps in the following sections minimize those weaknesses by some further filtering and reconstruction methods.

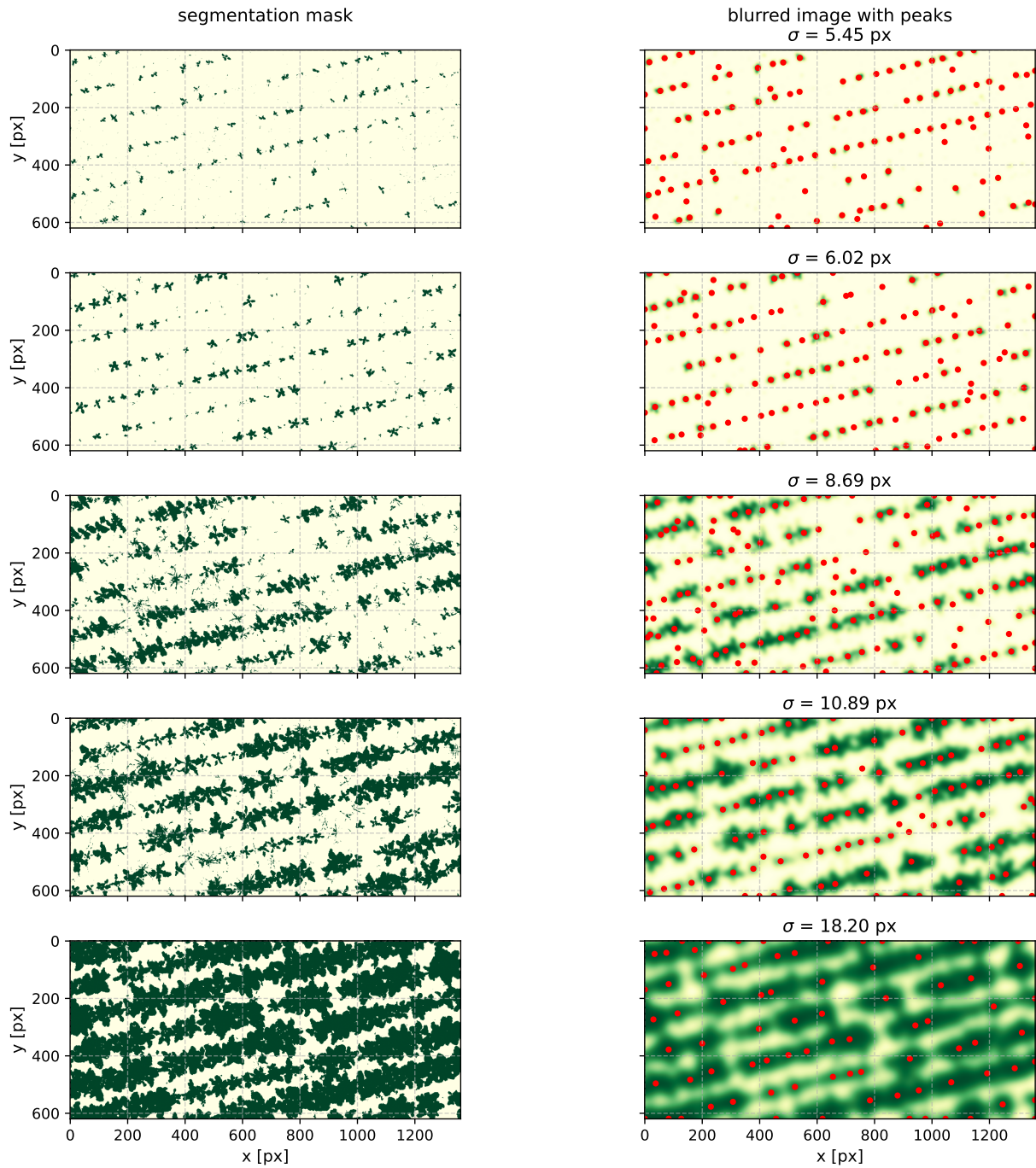


Figure 2.6: Plant peaks for different growing stages. The left plots show the determined segmentation masks for different growing stages of different fields. On the right, the results of the Gaussian blurring with the adaptive bandwidth and the peak detection (red dots) is shown.

2.4.3 Plant Group Registration

The steps described in Section 2.4.2 yield the peak positions for each valid—i.e. with sufficiently low cover ratio—UAV image as pixel coordinates. The goal of this next working step is to group the detected plant positions over time in order to identify single plants throughout the whole growing season which is, basically, a spatio-temporal clustering problem. The main challenge is that not every single plant is detected in each image, especially in later growth stages with high cover ratios. Additionally, there might be false positive detections due to weed or other objects in the field. Thus, a robust method is required to deal with these issues. The following sections will describe the particular steps of this grouping procedure.

Initially, by using the available geospatial information, the rasterized pixels can be translated into absolute GPS coordinates. Further, a universal transverse Mercator (UTM) coordinate system [135] is applied to translate longitude and latitude into a metric scale, e.g. centimeters. This yields metric peak positions

$$\mathcal{P}(t) = \{\vec{x}_i(t)\}_{i=1}^{N_p(t)}, \quad (2.9)$$

where $\vec{x}_i(t)$ is the position vector of the i -th peak and $N_p(t)$ is the number of detected peaks in the image with acquisition date t . Further, let $\mathcal{T}$ be the set of all acquisition dates and $\mathcal{T}^* \subseteq \mathcal{T}$ the set of all acquisition dates where the peak extraction method was performed. The goal of this section is to group this—yet independent—plant positions in a spatial way to identify single plants throughout all images of the whole growing season, even though they might not have been detected in every single image. Therefore, it is mandatory to align the image series in order to correct latent stitching errors or calibration glitches of the UAV’s GPS sensor. Additionally, fixed georeferenced points can be used to align the images to each other. If the plants are seeded in certain line structures, one can apply further steps by finding those lines and filtering “off-line” weed. This procedure is elaborated in the following.

2.4.3.1 Aligning Plant Positions

In order to avoid confusion of different plant IDs, the metric coordinates $\mathcal{P}(t)$ for each acquisition date t are aligned to each other. Not every single plant is detected in each image. Therefore, the point clouds visualizing the plant positions are not congruent but still highly correlated. Additionally, also weed and other objects that can be observed in the VI images may be included in $\mathcal{P}(t)$. However, these incorrectly detected objects are assumed stationary in location as well and, thus, are helpful with the alignment. Afterward, methods will be described to reduce unwanted objects from the detected peak positions. The main goal of this step is to avoid the largest GPS calibration errors. An

application example is shown in Figure 2.7. In the following, the particular processing steps are described. Furthermore, Algorithm 2.1 shows commented pseudocode for the position alignment.

Algorithm 2.1 Plant position alignment

Input:

- $\bar{\mathcal{P}}(t)$: centralized point clouds ▷ Eq. (2.10)
- $\mathcal{T}^*$: acquisition dates sorted by cover ratio in ascending order
- $t_{\text{init}} \in \mathcal{T}^*$: acquisition date of initial point cloud with the lowest cover ratio
- d_{register} : maximum next neighbor distance between basis and floating layer
- $d_{\text{group}} < d_{\text{register}}$: maximum distance from centroid for potential group members

Output:

- $\bar{\mathcal{P}}_{\text{aligned}}(t)$: aligned centralized point clouds
 - $\vec{R}(t) := [S(t), \alpha(t), B_0(t), B_1(t)]^\top$: transforms for each layer ▷ Eq. (2.12)
 - 1: initialize $\vec{R}(t_{\text{init}}) \leftarrow [1, 0, 0, 0]^\top$
 - 2: initialize $\bar{\mathcal{P}}_{\text{aligned}}(t_{\text{init}}) \leftarrow \bar{\mathcal{P}}(t_{\text{init}})$
 - 3: initialize $\bar{\mathcal{P}}_{\text{comb}} \leftarrow \bar{\mathcal{P}}(t_{\text{init}})$ ▷ point cloud of cluster centroids (initialized by first layer)
 - 4: **for** $t \in \mathcal{T}^* \setminus \{t_{\text{init}}\}$ **do**
 - 5: $\text{nn_model_comb} \leftarrow \text{nearestNeighborModel}(\bar{\mathcal{P}}_{\text{comb}})$ ▷ fit nearest neighbor model to grouped centroid layer
 - 6: $\vec{d}_{\text{new}} \leftarrow \text{nn_model_comb.evaluate}(\bar{\mathcal{P}}(t))$ ▷ get distances for all points of new layer to their next neighboring centroid
 - 7: $\text{nn_model_new} \leftarrow \text{nearestNeighborModel}(\bar{\mathcal{P}}(t))$ ▷ fit nearest neighbor model to new layer
 - 8: $\vec{d}_{\text{comb}} \leftarrow \text{nn_model_new.evaluate}(\bar{\mathcal{P}}_{\text{comb}})$ ▷ get distances for all centroids to their next neighboring point of the new layer
 - 9: $\vec{R}(t) \leftarrow \text{rigidCPD}(\{\bar{\mathcal{P}}_{\text{comb}} \mid d_{\text{comb}} \leq d_{\text{register}}\}, \{\bar{\mathcal{P}}(t) \mid d_{\text{new}} \leq d_{\text{register}}\})$ ▷ perform rigid coherent point drift (CPD) registration with subsets
 - 10: $\bar{\mathcal{P}}_{\text{aligned}}(t) \leftarrow \text{transform}(\bar{\mathcal{P}}(t), \vec{R}(t))$ ▷ transform complete set using Eq. (2.12)
 - 11: $\bar{\mathcal{P}}_{\text{comb}} \leftarrow \text{clusterPoints}(\bar{\mathcal{P}}_{\text{comb}} \cup \bar{\mathcal{P}}_{\text{aligned}}(t), d_{\text{group}})$ ▷ get cluster centroids (cf. Algorithm 2.3)
 - 12: **end for**
-

Currently, $\mathcal{P}(t)$ contains (absolute) metric coordinates. However, for the next steps, it is convenient to centralize the coordinates by subtracting the centroid of all peak positions. This results in the centralized coordinates

$$\begin{aligned} \bar{\mathcal{P}}(t) &= \{\vec{x}_i(t) - \vec{x}_{\text{mean}}\}_{i=1}^{N_p(t)}, \\ \vec{x}_{\text{mean}} &:= \frac{1}{|\mathcal{T}^*|} \sum_{t \in \mathcal{T}^*} \frac{1}{N_p(t)} \sum_{i=1}^{N_p(t)} \vec{x}_i(t). \end{aligned} \tag{2.10}$$

Generally, one can apply a transform containing constant shift, shearing, rotation, and scaling by an *affine* transform given by the rule

$$\mathbf{x}'(t) = \begin{bmatrix} T_{00}(t) & T_{01}(t) \\ T_{10}(t) & T_{11}(t) \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} B_0(t) \\ B_1(t) \end{bmatrix}, \tag{2.11}$$

where $\mathbf{x}(t)$ is the $(2 \times N_p(t))$ matrix of the centralized plant positions $\bar{\mathcal{P}}(t)$. Shearing effects are assumed irrelevant for the case of GPS calibration issues. By this assumption, one can concretize Equation (2.11) by only allowing shift, rotation, and scaling which results in

$$\mathbf{x}'(t) = S(t) \begin{bmatrix} \cos \alpha(t) & -\sin \alpha(t) \\ \sin \alpha(t) & \cos \alpha(t) \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} B_0(t) \\ B_1(t) \end{bmatrix}. \quad (2.12)$$

Thus, the generic transformation matrix elements $T_{ij}(t)$ reduce to a scaling factor $S(t)$ and a rotation angle $\alpha(t)$. Initially, the images should be aligned at least roughly so that one can assume $S(t) \approx 1$ and $\alpha(t) \approx 0^\circ$. The shift parameters $B_0(t)$ and $B_1(t)$ should be in the order of several cm. Now, a method usually called *registration* will be performed. Constraining the transform to the setting in Equation (2.12), it is also referred to as *rigid registration*. For the optimization, the method of coherent point drift (CPD) [98] is applied. Roughly speaking, it is based on a basis point set and a floating one, where the basis point set acts as data points. The floating-point set behaves as the centroids of a Gaussian mixture model (GMM). The objective is to minimize the negative log-likelihood (NLL) function. At the minimum, the two point sets are considered to be optimally aligned—or registered—to each other. More detailed information is provided in [98].

In order to align all (centralized) *point clouds* $\bar{\mathcal{P}}(t)$ corresponding to the acquisition dates, one has to apply the procedure multiple times for all layers in $\mathcal{T}^*$ step by step. Typically, the lower the cover ratio is, the more peaks are detected. One of the reasons is that the lower blurring bandwidth keeps finer structures visible for low cover ratios. Therefore, it is useful to sort the acquisitions not by calendar date but by ascending cover ratio as estimated before—either by the growth function fit (cf. Section 2.4.2.3), or by the cover ratio estimation (cf. Section 2.4.2.2). For rigid registration, a basis point set and a floating point set is needed. As an initial basis set, the point cloud belonging to the image with the lowest cover ratio with acquisition date $t_{\text{init}} \in \mathcal{T}^*$ is chosen, i.e. $\bar{\mathcal{P}}(t_{\text{init}})$. The floating set is the point cloud with the next higher cover ratio. In order to make the alignment robust, only those points of both sets are considered which have a next neighbor point in the other set within a maximum distance d_{register} .

After performing the registration, the resulting transformation (cf. Equation (2.12)) is applied on all points of the floating layer. The new basis set is formed by grouping the aligned points that are in the vicinity of each other and calculating their centroid. The resulting set $\bar{\mathcal{P}}_{\text{comb}}$ contains only those group—or cluster—centroids. Points without neighbors inside a given threshold distance d_{group} are considered new groups. This distance should be chosen smaller than d_{register} . Subsequently, the method is repeated

with the basis (centroid) set $\bar{\mathcal{P}}_{\text{comb}}$ and the point cloud with the next higher cover ratio as a floating set.

The developed grouping algorithm is applied in the final grouping as well, so a detailed discussion on it is done later in Section 2.4.3.4.

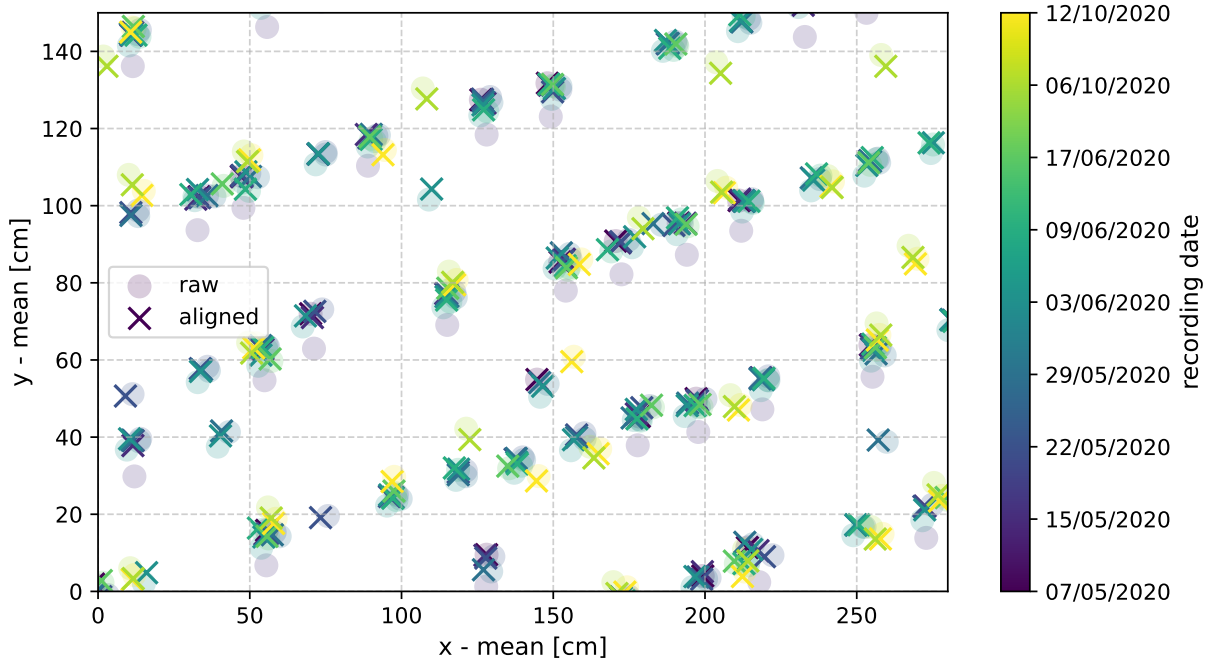


Figure 2.7: Point cloud alignment. Exemplary snippet from the application on peak positions. The color encodes the chronologically sorted acquisition date. Since the alignment procedure iterates over the layers with ascending cover ratio, this is not necessarily the alignment order. Shaded circles show the raw peak position data, whereas the crosses represent the same data after alignment. The data is already pre-aligned quite well so that only minimal improvements can be observed.

2.4.3.2 Recognizing Seeding Lines

The peak positions are aligned to each other, which means that possible image acquisition errors should be mostly eliminated. Usually—like in the given datasets—the plants are seeded in straight, parallel seeding lines. Recognized peaks off those lines can then be considered weed, other unwanted objects, or miscellaneous noise. Therefore, it is convenient to extract the seeding line positions in order to filter those out. In order to do this, the possible seeding lines are aligned with the x - or y -axis, since this is not necessarily the case for the GPS-referenced images which are aligned to a geographical coordinate system. In this work, the seeding lines are aligned to be parallel to the x -axis. In contrast to approaches like in [82] that use the image information directly, this approach only uses the point cloud information to infer the seeding line positions and angles. Algorithm 2.2 shows commented pseudocode for seeding line recognition.

Let us consider the acquisition dates $\mathcal{T}^*$ for which the peak point clouds are available. The idea of the seeding lines recognition is to apply a rotation transform on the

joint point clouds $\bigcup_{t \in \mathcal{T}^*} \bar{\mathcal{P}}_{\text{aligned}}(t) = \bar{\mathcal{P}}_{\text{aligned}}$. The task is to find the right rotation angle α_s . An established method in the field of computer vision to find regular structures like lines and circles in images is the *Hough transform* [33]. It detects points in images that form line structures by transformation of those lines into a feature space (Hough space) consisting of the line angle θ and its minimum distance d from origin. Thus, an infinite line in the (Euclidean) image space is mapped to a point in the Hough space. The brute force approach of filling the Hough space is to scan in each non-zero image point a bunch of lines with different angles that intersect in this point. If another point is on this line, one increments the corresponding (θ, d) -bin in Hough space. In the end, the infinite lines that are found in the image are visible by “nodes” in Hough space.

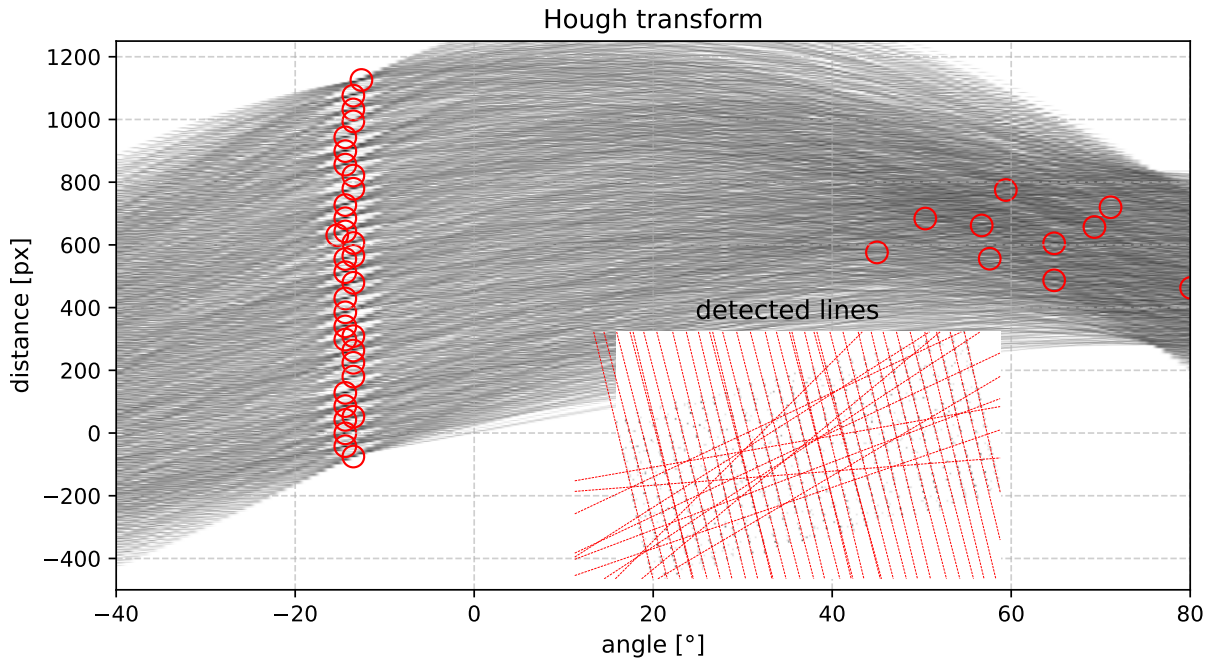


Figure 2.8: Seeding line rotation angle determination with Hough transform. The outer plot shows the Hough transform applied to the rasterized image shown in the inner plot. For reasons of visibility, the inner image is rasterized with a 5 times larger bin width than for the actual Hough transformed image. One can see a bunch of nodes in the Hough transform at angles of roughly -14° . Each detected node—marked with a red circle—corresponds to one (red) line in the inner image. By the vertical distance of the nodes, one can infer the line distances.

To use this method, $\bar{\mathcal{P}}_{\text{aligned}}$ needs to be transferred into a rasterized image by binning the peaks into a 2D histogram with a fixed bin width. Thus, the bins can be interpreted as image pixels. However, the gained image has a substantially lower resolution than the original image, which makes this method faster than considering the full-resolution plant images as in [82]. Subsequently, a fixed amount of angles in $(-90^\circ, 90^\circ]$ is scanned. This results in a rather rough search of angles, although all seeding lines are expected to be rotated with a common angle. Other randomly found lines in the image can occur, but they are at diffuse angles. The correct seeding line angle is expected to occur regularly so that a clustering around the correct seeding line angle is expected.

Thus, the principle of nested intervals is used: the interval of queried angles is first divided into small bins. In the next step, the histogram of all found angles is calculated by using those bins. The diffuse angles will distribute over multiple bins, whereas the correct seeding line angle should accumulate in one or at least few bins. The accumulation bin and maybe the surrounding bins as well are then taken as the new query interval which is then again divided into bins. After several iterations, one finds the common angle of all visible seeding lines α_s . Figure 2.8 shows the Hough transform applied on a rasterized peak position image.

Since the Hough transform not only yields the line angles but also their distances, one might use this method directly to extract the seeding lines itself. However, it may happen that the Hough transform-based detection misses some less prominent line constellations so that the distances may be only multiples of the true seeding line distance. In order to have a more stable procedure, the seeding line positions are searched again in the point cloud, which is rotated by α_s . Nevertheless, one can use the distance information of the Hough transform results as an estimate for the expected seeding line distances. To have a stable estimate which is robust against missed seeding lines, the median seeding line distance is used for the next step.

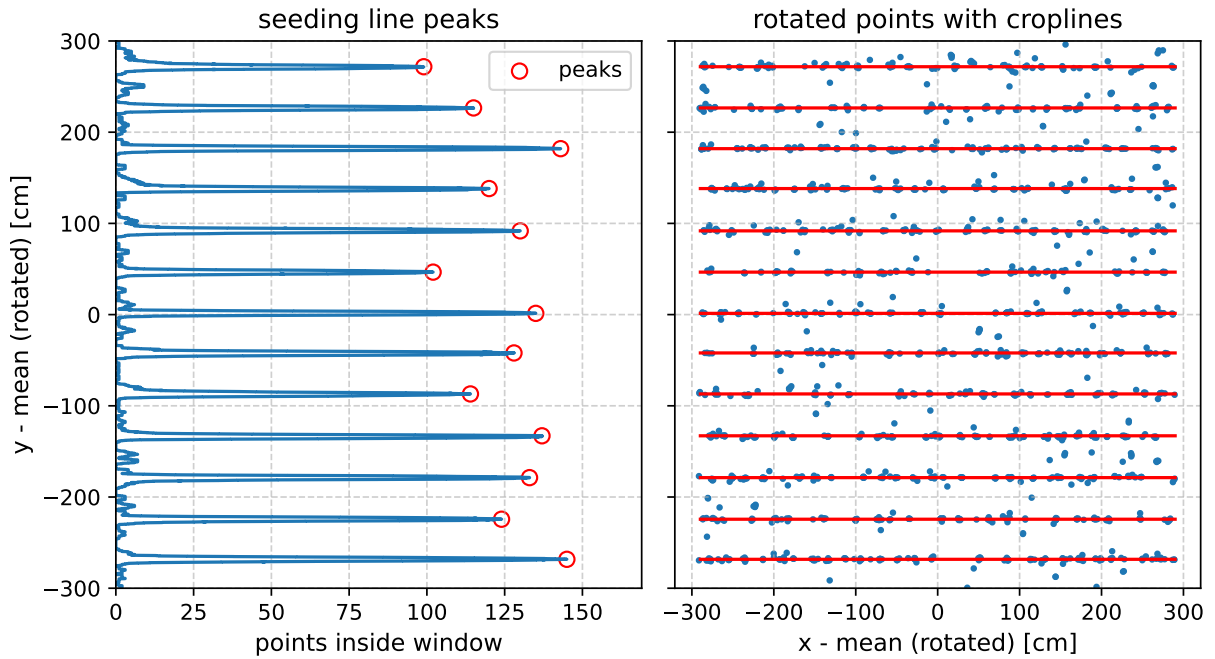


Figure 2.9: Seeding line recognition. For the seeding line recognition, the points inside a window with the size $\lambda = 32$ px are scanned with a precision of $\rho = 0.5$ px as seen in the left plot. In this example, 29 valid peaks are found by k-means. They represent the center y -coordinates of the seeding lines shown in the right plot with the corresponding (rotated) peak positions.

Only the y -coordinates $\vec{y}_s$ of the rotated point cloud are considered. The principle is to count the points that are inside a certain window with a central y coordinate and a size λ . A scan through all coordinates with a given precision ρ , i.e. the distance between

two nearest window centers, results in the sum of included points against the window center position. Finally, one applies a peak finding method to extract the local maxima which represent the (sorted) seeding line positions $\vec{y}^*$. An example plot can be found in Figure 2.9. The success of this method is strongly dependent on the right choice of λ and ρ . In order to get this procedure to work stable, it is helpful to set λ and ρ in relation with the seeding line distance estimate of the Hough transform. The peak finding can also be tweaked considering that one actually knows roughly where the peaks should be located.

Algorithm 2.2 Seeding line recognition**Input:**

- $\mathcal{T}^*$: acquisition dates with given peak positions
 $\mathcal{Q} := \bigcup_{t \in \mathcal{T}^*} \bar{\mathcal{P}}_{\text{aligned}}(t)$: centralized aligned point clouds
 $\mathbf{q} \in \mathbb{R}^{2 \times n}$, $n = |\mathcal{Q}|$: matrix of point coordinates in point cloud $\mathcal{Q}$ with $\vec{q}_i = [x_i, y_i]^\top$
 n_b : number of bins for Hough angles histogram
 n_+ : number of surrounding bins to respect for nesting iteration
 λ : window length for seeding line position scan
 ρ : precision for seeding line position scan

Output:

- $\vec{y}^*$: seeding line positions
 α_s : rotation angle
- 1: initialize angle interval $[\alpha_{\min}, \alpha_{\max}] \leftarrow [-90^\circ, 90^\circ]$
 - 2: $\mathbf{I} \leftarrow \text{2d_histogram}(\mathbf{q})$ $\triangleright$ raster point coordinates into image with given bin width
 - 3: **repeat**
 - 4: $\mathbf{H} \leftarrow \text{hough_transform}(\mathbf{I}, [\alpha_{\min}, \alpha_{\max}])$ $\triangleright$ perform Hough transform on image in given interval of angles
 - 5: $\vec{d}, \vec{\alpha} \leftarrow \text{find_lines}(\mathbf{H})$ $\triangleright$ find nodes in Hough image yielding line distances $\vec{d}$ and corresponding angles $\vec{\alpha}$
 - 6: $\mathcal{B} \leftarrow$ set of n_b equal binned intervals in $[\alpha_{\min}, \alpha_{\max}]$
 - 7: $\vec{h} \leftarrow \text{histogram}(\vec{\alpha}, \mathcal{B})$ $\triangleright$ $\vec{h}$ counts the number of nodes in the corresponding bin in $\mathcal{B}$
 - 8: $\mu = \frac{1}{\dim \vec{\alpha}} \sum_{i=1}^{\dim \vec{\alpha}} \alpha_i$ $\triangleright$ take the interval, where most of the nodes are included
 - 9: $[\alpha_{\min}, \alpha_{\max}] \leftarrow \bigcup_{i=\max\{1; \arg \max \vec{h} - n_+\}}^{\min\{n_b; \arg \max \vec{h} + n_+\}} \mathcal{B}_i$
 - 10: **until** $\mu = \alpha_i \forall \alpha_i \in \vec{\alpha}$
 - 11: $\alpha_s \leftarrow \alpha_{\min}$
 - 12: $\mathbf{q}_s \leftarrow \begin{bmatrix} \cos \alpha_s & -\sin \alpha_s \\ \sin \alpha_s & \cos \alpha_s \end{bmatrix} \mathbf{q}$ $\triangleright$ rotated point cloud defined analogously to $\mathcal{Q}$, thus $\vec{q}_{s,i} = [x_{s,i}, y_{s,i}]^\top$
 - 13: initialize point sum vector $\vec{\sigma} \leftarrow \vec{0}$
 - 14: initialize $i \leftarrow 0$
 - 15: initialize $y_{\text{test}} \leftarrow \min \vec{y}_s - \lambda$
 - 16: **while** $y_{\text{test}} < \max \vec{y}_s + \lambda$ **do**
 - 17: $\sigma_i \leftarrow \sum_{l=1}^{\dim \vec{y}_s} I_{\{y_{\text{test}} - \lfloor \frac{\lambda}{2} \rfloor \leq y_{s,l} < y_{\text{test}} + \lfloor \frac{\lambda}{2} \rfloor\}}$
 - 18: $y_{\text{test}} \leftarrow y_{\text{test}} + \rho$
 - 19: $i \leftarrow i + 1$
 - 20: **end while**
 - 21: $\vec{p}_{\text{peaks}} \leftarrow \text{peakfinder}(\vec{\sigma})$ $\triangleright$ finds local maximum peak positions representing seed line positions
 - 22: $\vec{y}^* \leftarrow (\min \vec{y}_s - \lambda) \vec{1} + \rho \vec{p}_{\text{peaks}}$ $\triangleright$ $\vec{1} := [1, \dots, 1]^\top$

2.4.3.3 Filtering Weed

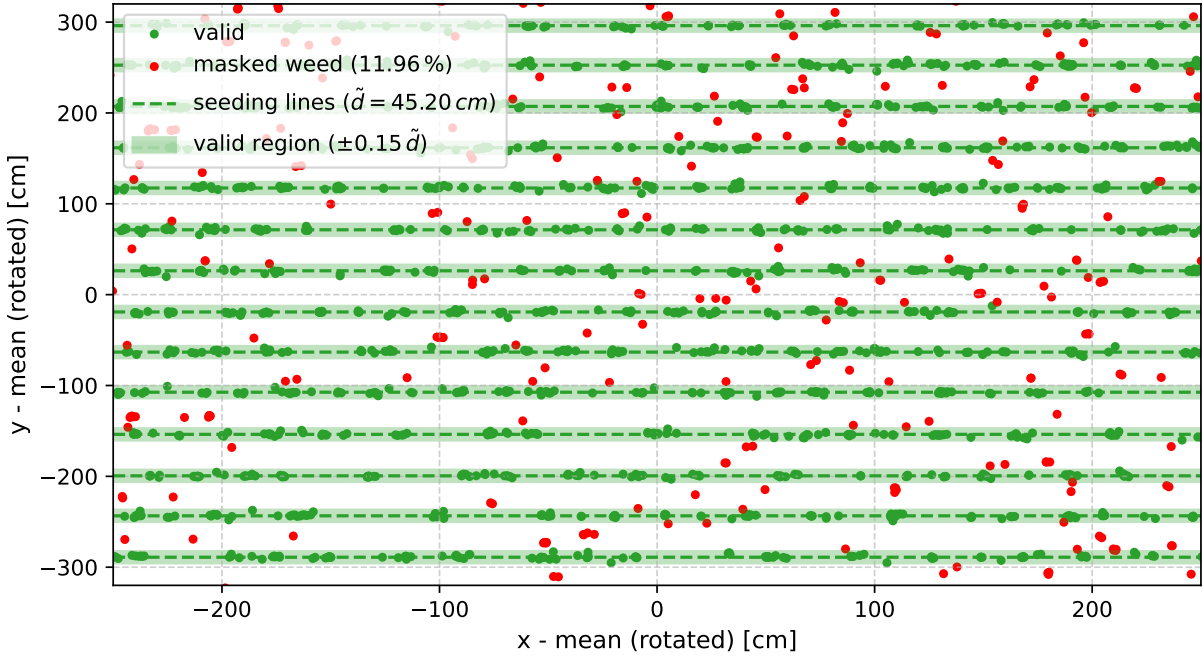


Figure 2.10: Weed filtering. Excerpt from peak positions where weed filtering is applied. Peaks inside the green shaded valid regions are considered valid plants (green dots) whereas peaks being outside are masked as weed (red dots).

One benefit of knowing the seeding line positions is that one can effectively filter out recognized objects that are located off the lines which are mainly weed or false detections. Usually, the seeding lines have a regular distance to their neighboring ones so that all distances are approximated by their mean distance. However, if the images contain few irregular seeding line distances, e.g. due to cart tracks in between, it is again helpful to prefer the median distance $\tilde{d}$ over the mean. Additionally, for the distance of each point to its next seeding line, it is

$$d_i = \min |y_i \vec{1} - \vec{y}^*|, \quad (2.13)$$

where y_i is the centralized, aligned, and rotated y -coordinate of the i -th plant position and $\vec{1}^\top = \{1\}^{\dim \vec{y}^*}$. Once all nearest distances d_i are determined, a threshold factor ϑ_d can be set that specifies which proportion of the median distance $\tilde{d}$ should be the maximum distance of each plant position to their next seeding line to be considered valid detection. Hence, the condition is

$$d_i \leq \vartheta_d \tilde{d}. \quad (2.14)$$

The threshold factor is another parameter of the workflow that has to be chosen deliberately. Too small values lead to the incorrect identification of plants as weed, while too

large values reduce the filter effect so that much weed is recognized as plants. Figure 2.10 shows an example weed filtering with threshold factor $\vartheta_d = 0.15$ which turned out to be a good choice for the considered datasets.

2.4.3.4 Linking Individual Plants during Time

After aligning and filtering the peak positions, $\mathcal{P}(t)$ are redefined to be the aligned, filtered plant positions and $\mathcal{P} := \bigcup_{t \in \mathcal{T}^*} \mathcal{P}(t)$ to be the unified point cloud.

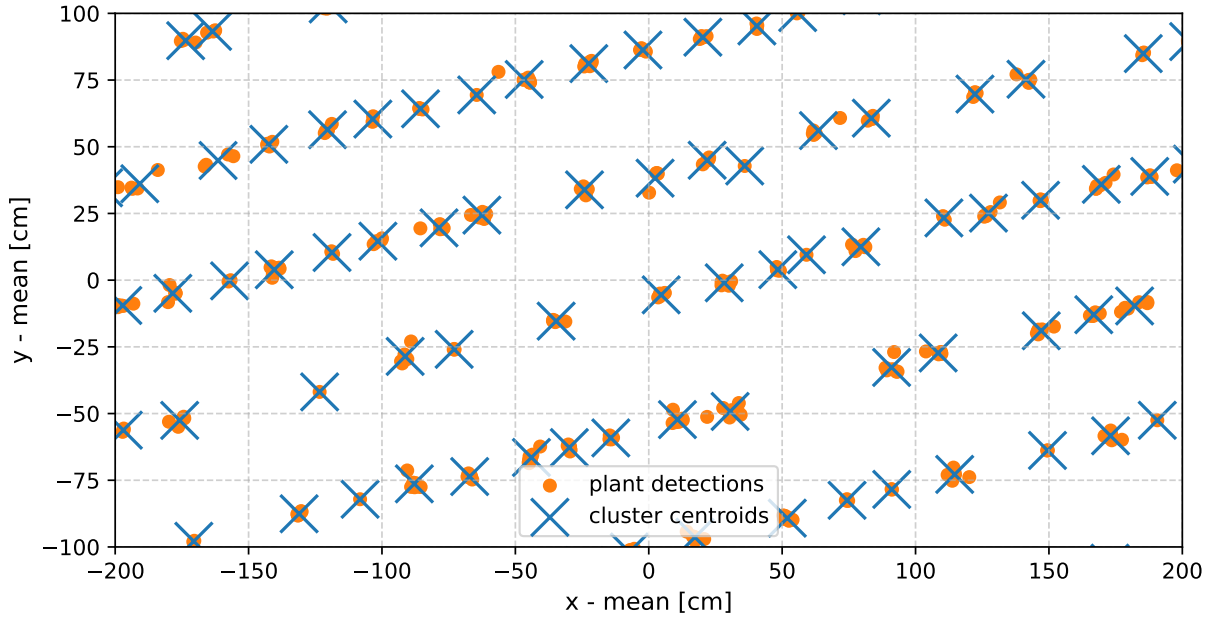


Figure 2.11: Cluster centroids. Each cluster represents a plant and incorporates the detections in the image series.

Furthermore, the corresponding image acquisition dates of all plants are summarized in the label vector $\vec{t} \in (\mathcal{T}^*)^{|\mathcal{P}|}$. The goal is to group the point cloud into clusters $\mathcal{G} \subset \mathcal{P}$ and thereby identify single plants recorded in those images where they were detected by the previous methods. An important coordinate of each cluster therefore is its centroid $\vec{\zeta}$ calculated by

$$\vec{\zeta}_i = \frac{1}{|\mathcal{G}_i|} \sum_{\vec{x} \in \mathcal{G}_i} \vec{x}, \quad (2.15)$$

where the $\vec{x}$ are the clusters' point coordinates in $\mathcal{P}$. Moreover, let $\mathcal{C} := \{\vec{\zeta}_i\}_{i=1}^{n_c}$ be the set of all n_c cluster centroids. One can postulate that each cluster contains not more than one member per acquisition date in $\mathcal{T}^*$. Using classical approaches like k -means or even more sophisticated ones like density-based spatial clustering of applications with noise (DBSCAN) [35] for the clustering, one can hardly make use of this knowledge.

Those common methods only handle single point clouds without integrating more information on those points. Thus, there may be multiple points from the same acquisition date in one cluster. To integrate the acquisition date information, an iterative—or “layer-wise”—approach comparable to a point-to-point registration is used; similar to the CPD method (cf. Section 2.4.3.1). The different acquisition dates are considered layers of points. Plant detections in plots with low cover ratio are more precise in general, so the layering starts from the layer of the date with the lowest cover ratio, as for the alignment. Each plant position is considered to be the centroid of a cluster containing—so far—one member. Each centroid is assigned a unique cluster ID. Subsequently, the layer with the second-lowest cover ratio is considered. A nearest neighbor model based on the cluster centroids is queried to get the Euclidean distances for each point in the new layer to its next cluster centroid. Next, some decisions have to be made based on this next neighbor distance d . A maximum point-centroid distance $d_{\max}$ for points to be considered members of the corresponding clusters is set. Thus, if the new candidate is inside this distance ($d \leq d_{\max}$), those points are assigned with the corresponding cluster ID. It can happen that multiple candidates are inside this distance. Then, only the closest point is registered to the cluster and all other candidates are discarded. Points outside this distance ($d > d_{\max}$) are assumed to be new clusters labeled new individual cluster IDs. Finally, the cluster centroids are recalculated by assigning the new candidates to their corresponding clusters. This procedure is repeated iteratively until all layers are processed. In the end, one gets clusters with at most $|\mathcal{T}^*|$ members and a label vector $\vec{l}$ containing the corresponding cluster ID for each point. Figure 2.11 shows an exemplary plot of the cluster centroids that now represent single plants. The IDs are ascending integer values starting from 0. Discarded points get the label -1 . Algorithm 2.3 shows the linking procedure as commented pseudocode.

Algorithm 2.3 Iterative point clustering**Input:**

- $\mathcal{P}$: aligned point clouds
- $\mathcal{T}^*$: acquisition dates with plant position information sorted by cover ratio in ascending order
- $t_{\text{init}} \in \mathcal{T}^*$: acquisition date of initial point cloud with the lowest cover ratio
- d_{max} : maximum distance from centroid for potential cluster members

Output:

- $\vec{l}$: label vector of cluster ID for each point
- $\mathcal{C}$: cluster centroids

- 1: initialize $\vec{l} \leftarrow \{-1\}^{|\mathcal{P}|}$ ▷ -1 = not assigned to any cluster
- 2: $\{\vec{l} \mid t = t_{\text{init}}\} \leftarrow \{0, 1, \dots\}$ ▷ assign individual labels for each point of first layer
- 3: initialize $\mathcal{C} \leftarrow \{\mathcal{P} \mid t = t_{\text{init}}\}$
- 4: **for** $t' \in \mathcal{T}^* \setminus \{t_{\text{init}}\}$ **do**
- 5: $\text{nn_model} \leftarrow \text{nearestNeighborModel}(\mathcal{C})$ ▷ fit nearest neighbor model
- 6: $\vec{d}, \vec{\tilde{l}} \leftarrow \text{nn_model.evaluate}(\{\mathcal{P} \mid t = t'\})$ ▷ get next neighbor distance and ID for all points of respective layer; next neighbor ID corresponds to cluster ID
- 7: initialize $\vec{m} \leftarrow \{1\}^{\dim \vec{d}}$ ▷ vector for new cluster member candidates
1 = valid member candidate for existing cluster
0 = valid member candidate for new cluster
-1 = excluded due to multiple candidates for a single cluster
- 8: $\{m \in \vec{m} \mid d > d_{\text{max}}\} \leftarrow 0$ ▷ assign points with no existing cluster in their vicinity to a new cluster ($m \leftarrow 0$)
- 9: **for** $\tilde{l}' \in \vec{\tilde{l}}$ **do** ▷ iterate over the set of cluster IDs in $\vec{\tilde{l}}$
- 10: $\{m \in \vec{m} \mid d \neq \min\{d \in \vec{d} \mid \tilde{l} = \tilde{l}'\}\} \leftarrow -1$ ▷ for multi-assignments: get candidates for one cluster and keep the one with the minimal cluster distance; exclude the others ($m \leftarrow -1$)
- 11: **end for**
- 12: $\{l \in \vec{l} \mid t = t' \wedge m = 1\} \leftarrow \{\tilde{l} \in \vec{\tilde{l}} \mid m = 1\}$ ▷ assign valid member candidates for existing clusters to respective ones
- 13: $\{l \in \vec{l} \mid t = t' \wedge m = 0\} \leftarrow \{\max(\vec{l}) + 1, \max(\vec{l}) + 2, \dots\}$ ▷ give new cluster labels to valid member candidates for new clusters
- 14: $\mathcal{C} \leftarrow \emptyset$ ▷ reset clusters before recalculation
- 15: **for** $l' \in \vec{l} \setminus \{-1\}$ **do** ▷ iterate over the set of valid cluster IDs in $\vec{l}$
- 16: $\mathcal{G} \leftarrow \{\mathcal{P} \mid l = l'\}$ ▷ consider single cluster with cluster ID l'
- 17: $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ \frac{1}{|\mathcal{G}|} \sum_{\vec{x} \in \mathcal{G}} \vec{x} \right\}$ ▷ recalculate cluster centroids with new cluster assignments (cf. Eq. (2.15))
- 18: **end for**
- 19: **end for**

2.4.3.5 Further Filtering and Indirect Detections

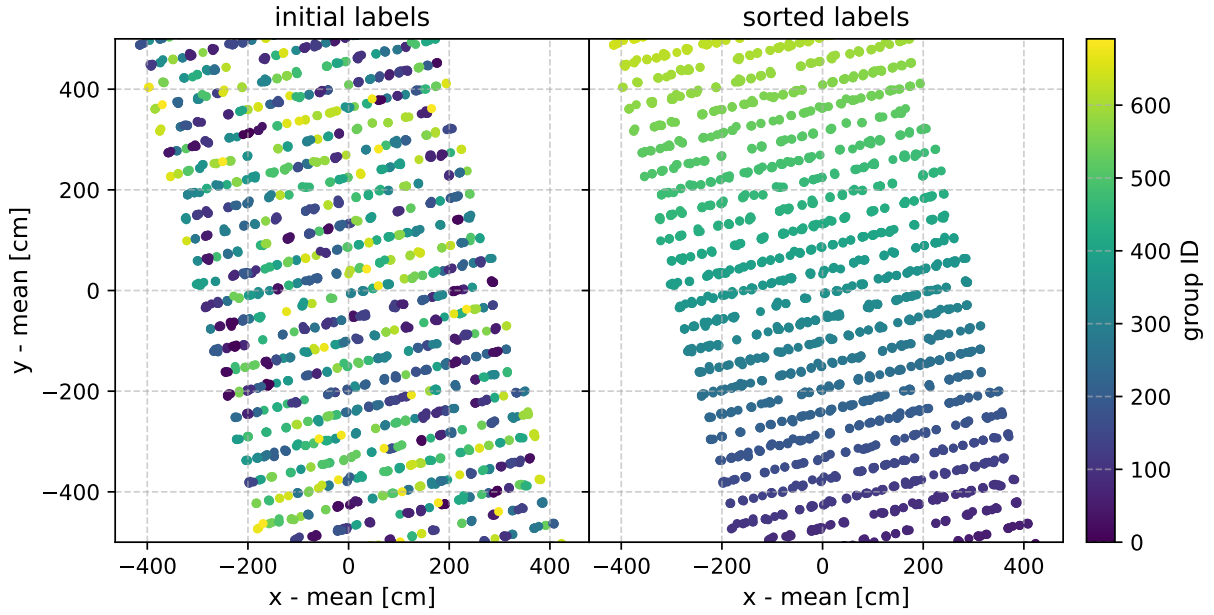


Figure 2.12: Group label sorting. By using the rotation-corrected coordinates, the plants (or plant groups) can be re-labeled with sorted labels. The color encodes the group ID.

With the information of the cluster centroids $\mathcal{C}$ from the results of Section 2.4.3.4, one can find a single plant also in such images where it is not necessarily detected since the spatial location is fixed. But first, a somehow “cosmetic” step is done.

The label numbering resulting from the clustering algorithm is continuous but—due to new clusters starting at different steps—not sorted. However, one may want to have the labels sorted for reasons of clarity and easier retrieval of the plants in in-field operations. A reasonable approach would be to use the seeding line coordinates $\vec{y}^*$ from Section 2.4.3.2 again in order to determine the seeding line ID for each plant group. The cluster centroids $\mathcal{C}$ are used to calculate the nearest seeding line and assign its ID i_s to each cluster.

$$i_s = \arg \min_{y^* \in \vec{y}^*} |y_c - y^*|, \quad (2.16)$$

where y_c is the y -coordinate of the cluster centroids. Thus, the initial labels are substituted with new sorted labels: primarily sorted by the seeding line ID and secondarily by—in this case for instance—the x -coordinates of cluster centroids. A figure that shows this label sorting for a complete field is shown in Figure 2.12.

Moreover, the special point in having the cluster centroid positions is that one can retrieve the plant positions in the high-cover-ratio images—those with acquisition dates $t \in \mathcal{T} \setminus \mathcal{T}^*$ —where above methods could not be applied. Additionally, not every plant may be recognized in all analyzed images. In this context, the detections by the above

methods are named *direct detections*, whereas the reconstructed positions by using the cluster centroids are referred to as *indirect detections*. The aligning algorithm not only gives us the aligned coordinates of the (directly detected) plant positions, but also the transform vectors $\vec{R}(t) := [S(t), \alpha(t), B_0(t), B_1(t)]^\top$ for each image³. Thanks to the alignments being (rigid) affine transforms, they are invertible. Hence, it is

$$\mathbf{x}(t) = \frac{1}{S(t)} \begin{bmatrix} \cos \alpha(t) & \sin \alpha(t) \\ -\sin \alpha(t) & \cos \alpha(t) \end{bmatrix} \left(\mathbf{x}'(t) - \begin{bmatrix} B_0(t) \\ B_1(t) \end{bmatrix} \right) \quad (2.17)$$

by rearranging Equation (2.12). Thus, one can calculate the plant positions for indirect detections with this inverse transform in order to fill up the point clouds. In order to recover the directly detected plant positions in the individual images, the calibration error (cf. Section 2.4.3.1) is added again. Please note further that $\mathbf{x}(t)$ are still the centralized UTM coordinates introduced in Section 2.4.3.1. For a complete inverse transform into the GPS coordinate system and the rasterized image pixel system, respectively, one has to add the mean vector $\vec{x}_{\text{mean}}$ again. Finally, the indirect detections at each acquisition date are basically the detected cluster centroids inversely transformed into the coordinate systems of the image data at respective acquisition dates.

All these above filtering rules result in a labeled point set where each cluster has exactly $n_t = |\mathcal{T}|$ members, one for each acquisition date $t \in \mathcal{T}$. Finally, the plants are claimed to be “cataloged”.

2.5 Evaluation and Discussion

In this section, the evaluation of the results on available ground truth information for both datasets is discussed. Furthermore, an exemplary application use case of the cataloging workflow for in-field operations is briefly presented.

With available manually annotated information on the plant positions as a reference, one can validate the above methods. For the sugar beet dataset, there is some ground truth data available which enables us to perform an evaluation of the binary classifiers of *true positive* (TP), *false positive* (FP), and *false negative* (FN) detections. Since true negative detections make no sense in this application, measures like the accuracy are not appropriate. Therefore, two other measures called *precision* and *recall* defined by

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.18)$$

³... as long as the cover ratio is not too high. If it is, however, the alignment cannot be performed and the transform vector is the identity transform $\vec{R}(t) := [1, 0, 0, 0]^\top$.

are used. Descriptively, the recall describes which ratio of actually seeded plants the cataloging workflow catches, whereas the precision measures the ratio of how many of its detections are indeed real plants. Dot-like positions should be compared to each other. Thus, a tolerance radius of maximum distance between a true plant position and its potential position of detection is set. This evaluation requires the plants to be recognized with a maximum tolerance of 8 cm—i.e. the maximum distance between detected plant position and ground truth annotation. The counting of TP, FP, and FN is done thanks to a nearest neighbor approach. For each plant detection the ID and the distance to the next true plant position is evaluated. One iterates over the true plant positions and for each of them, the distances from the plant detections that were assigned to those corresponding true plants are gathered. If there is at least one assignment inside the tolerance, TP is incremented by one and FP by the number of remaining total assignments. If no assignment is inside the tolerance, FN is incremented by one and—again—FP by the number of total assignments. Since every recognized plant position is assigned to exactly one true position, one can consider each true and recognized plant by this method. In the end, the binary classifier counts are obtained and precision and recall can be calculated by Equation (2.18). Moreover, this method yields the plant positions by direct and indirect detections. In the ground truth data, the plant is not annotated if it is not visible, for instance in an early image. Since the cataloging uses the inverse transformations of the plant position centroids $\mathcal{C}$, it would already find the plant even if it is not yet visible. Therefore, all “leading” indirect detections—i.e. all that are earlier than the first direct detection in terms of acquisition date—are ignored.

The upper plot of Figure 2.13 shows evaluations of precision vs. recall for field images with available ground truth. Below, for each acquisition date (encoded by the color), an example image is shown. For most of the images, a precision of at least 90 % is achieved. The recall is at least 90 % for most of the images. As expected, the plant detection gets slightly more inaccurate, the higher the cover ratio is due to the lower point-like shape of the plants. However, the early plant detections at lower cover ratios are more confident. This should give the hint that high-cover-ratio images are inappropriate for plant detection and should only be used for the indirect retrieval of already known plant positions at the cluster centroids $\mathcal{C}$.

Generally, the precision is lower than the recall, indicating that the workflow has some false positive plant recognitions. Mostly, those false plants are double recognitions or still remaining weed inside the seeding lines or within the tolerance region defined in Section 2.4.3.3. Since the cataloging is not sensitive to the actual plant shape like possible deep learning approaches, this remains to be a problem. The workflow tends to have a higher precision the better the fields are weeded. The on-line weed may be filtered manually or by further shape-sensitive methods, whereas double detections may be reduced by better clustering parameters.

Furthermore, evaluation of the cauliflower dataset used in [A] shows that the proposed workflow detects the plant positions more accurately than plant detections based on a deep learning approach with a Mask region-based convolutional neural network (R-CNN) especially considering the exact plant center positions. Thus, it is shown that the proposed workflow is not only more accurate, but also more generalizable by not being dependent on training data. The full results of the cauliflower dataset evaluation are presented in [A].

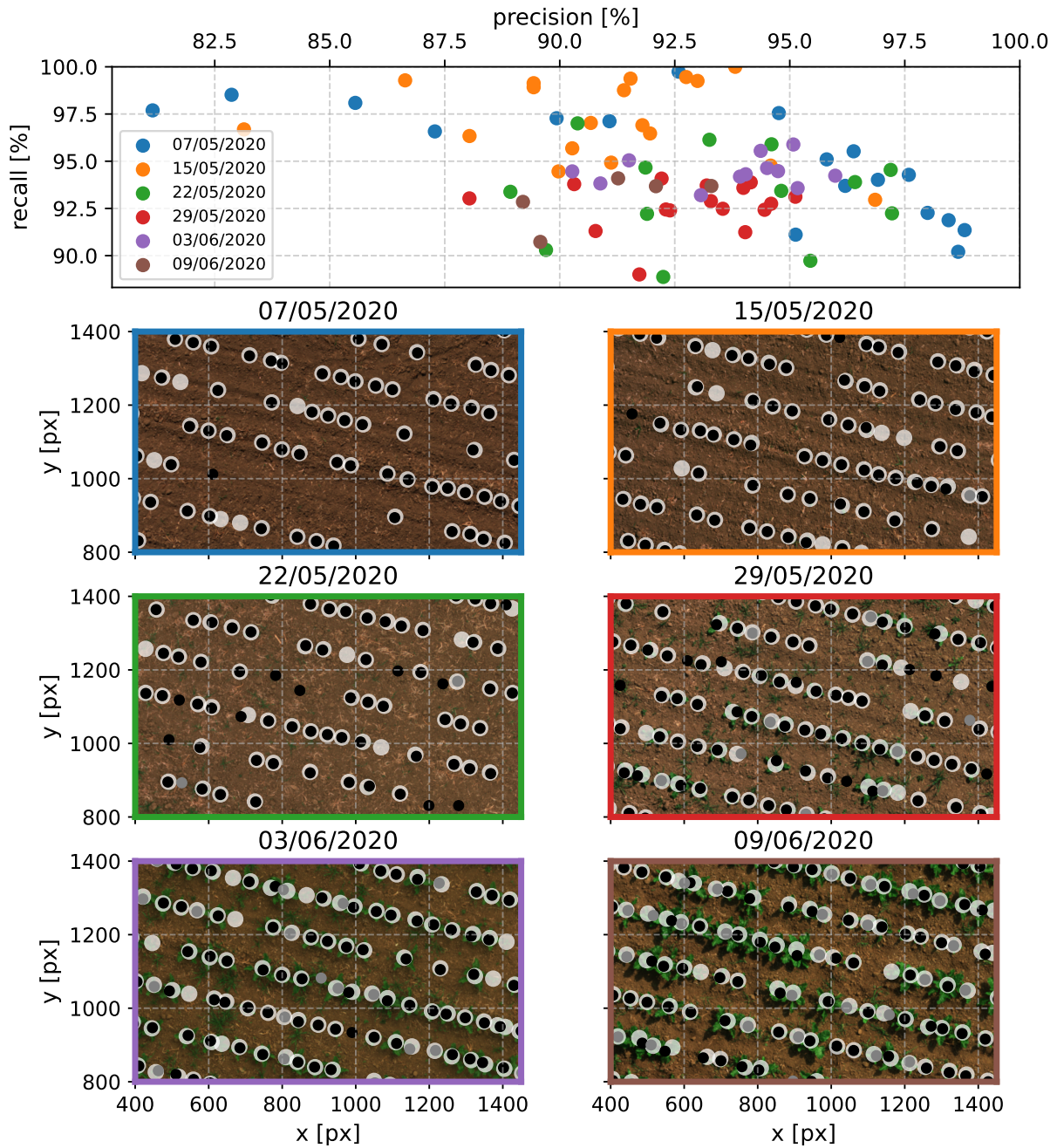


Figure 2.13: Precision and recall evaluation summary. The upper plot shows precision vs. recall for all field images with available ground truth data. Colors show the acquisition date of the corresponding image. Below, field excerpts are shown exemplarily—one for each available acquisition date. In these plots, the real plants are marked by white dots. Black dots represent direct detections via the peak detection method. The gray dots are indirect detections by using information of other images of the same field and back-transform the positions in each image.

2.5.1 Application to In-Field Annotations

For all plant positions, GPS coordinates are available which enables processing the data with geographic information system applications like *QGIS* [114]. For in-field operations like plant assessment or annotation, the plant catalog can support by providing a structured position sample of interesting plants. Additionally, there are some interesting applications like *QField* [108] that are able to process georeferenced data directly on a smartphone. This is an excellent tool for farmers, who then can do in-field tasks directly “online” using the plant catalog and their current GPS position. Exporting the plant catalog for instance as a keyhole markup language (KML) file [45] allows enriching the georeferenced plant catalog with further annotation data like disease severity score, number of leaves, additional images, etc. Figure 2.14 shows how this could possibly look like for an in-field scenario.

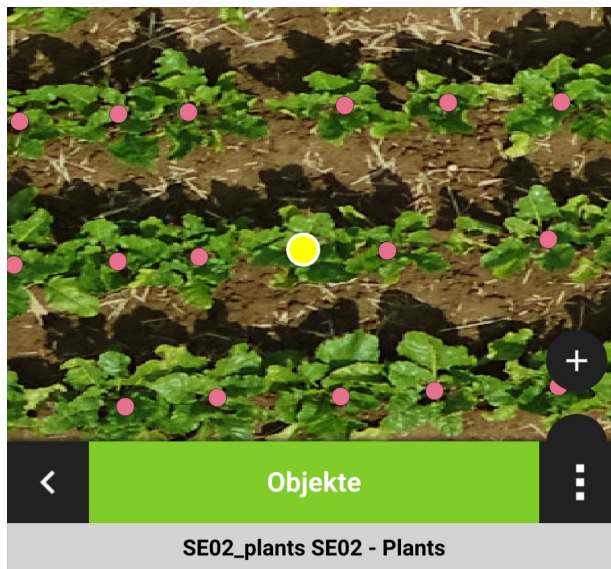


Figure 2.14: Plant catalog for in-field observations with QField. Exemplary screenshot on an Android smartphone. Detected plants can be accessed individually and various annotations like quality assessments, additional notes, or other images can be added.

2.6 Interim Conclusion

The presented workflow shows a very satisfactory detection performance for both considered datasets. By choosing RGB-only VIs for the peak detection, the cataloging demonstrates successfully that RGB information is sufficient. This could be important for use cases where a multi- or even hyperspectral data acquisition is not feasible for some reason. Nevertheless, it cannot be ruled out that the peak detection performance could be improved by having data beyond the optical spectrum available. The high degree of automatization enables the analysis of large-scale data. Particularly for those datasets consisting of multiple smaller fields, like in the sugar beet dataset, many workflow steps can be processed in parallel.

Deep learning based plant recognition approaches have to be trained on the distinct plant sizes if training data is available at all. Unfortunately, for the images with canopy closure ($c > 75\%$), both deep learning and the rather “classical” approach of plant cataloging will fail, since neighboring plants cannot be distinguished from each other anymore. However, for feasible plant sizes, the proposed cataloging approach is mostly

agnostic to the plant size and can effectively extract images that can be used to train even better deep learning based plant recognition systems.

By exploiting the geospatial information in UAV images, one can convert between pixel units and metric units like centimeters. From the user's point of view, this makes the workflow very adaptable to other datasets with different plants and seeding conditions: it allows to directly refer to real-world measures like plant radii or seeding distance without having to consider the image resolution.

Large deep learning models that are developed nowadays require large amounts of data, especially for tasks involving image processing like, e.g., classification and instance segmentation with Mask R-CNNs. In principle, non-invasive remote sensing (RS) and UAVs enable the application to large-scale agricultural experiments and corresponding data. By automatizing the plant cataloging and providing a data framework, the workflow proposed in this chapter helps to exploit the full potential of UAV imaging in agricultural contexts.

The plant catalog gathered with this image extraction workflow makes it possible to accumulate a time series of images of individual plants. By inverse transformation of the cluster member coordinates in their respective image pixel-based coordinate system (cf. Section 2.4.3.5) one gets the, say, "original" pixel positions of the plants. Next, one simply defines a frame around those pixels and receives smaller "tiles". These can then be used for further steps, for instance as a training set for neural network (NN) architectures as in [G]. The fact that the images are linked to individual plants at multiple dates enables this dataset to be used not only for spatially related but also for time series analyses. Figure 2.15 shows a few example image tiles for both investigated datasets.

The results of this work build the basis for further advanced image-based plant phenotyping applications as presented in the following chapter.

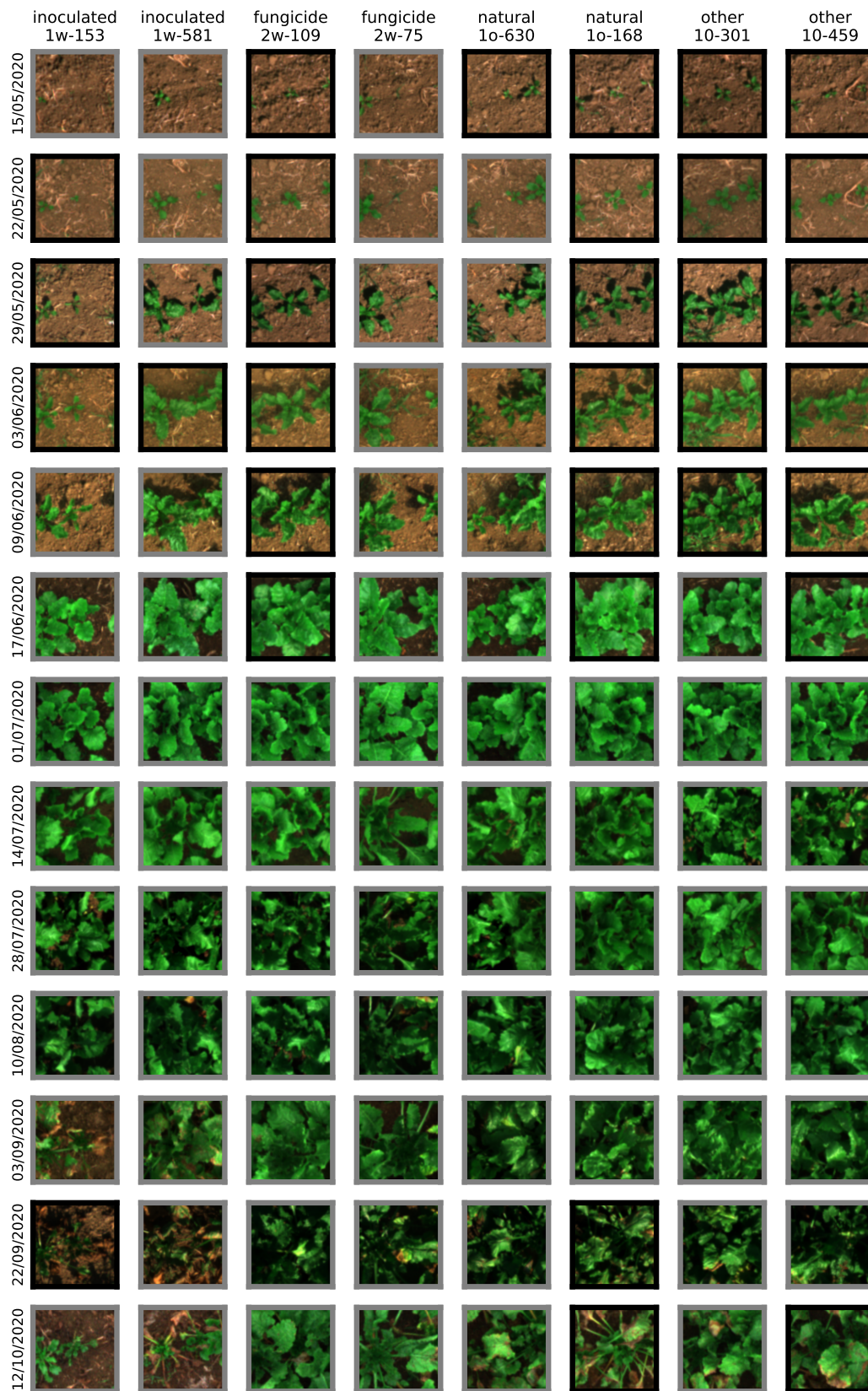
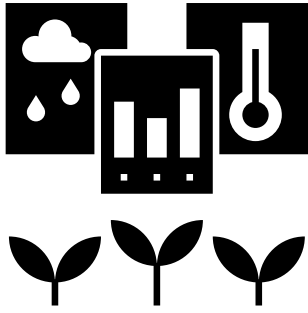


Figure 2.15: Plant image time series. Randomly picked image series of plant RGB images detected by the proposed cataloging workflow. The acquisition date increases downwards. Roughly every second acquisition date is shown. Black frames mark direct, gray frames indirect detections. The columns are ordered in blocks of two examples each from inoculated, fungicide-treated, natural and reference fields, respectively.



3

Disease Severity Prediction

The content of this chapter is based on the publication [E]. As the work presented in this publication is closely related to the plant cataloging workflow presented in Chapter 2, my contributions in this publication included the conceptualization of the disease severity prediction model, the design and implementation of the software, conducting the experiments, and writing the manuscript. The publication is a collaboration with colleagues from the Institute of Sugar Beet Research (IfZ) in Göttingen, where the data was acquired. The developed model is now used by the IfZ for automated disease severity annotation in sugar beet field experiments.

3.1 Introduction

In this chapter, the focus is on the automated annotation of disease severity (DS) in *Cercospora* leaf spot (CLS)-infected sugar beet fields. The in-field annotation by human experts is a time-consuming and labor-intensive task. Additionally, the annotation process is prone to subjective biases and inconsistencies because annotation references are often limited to example visualizations of plants for certain DS levels on which the annotators can rely. To get a high annotation throughput and to reduce the subjectivity, this chapter proposes a framework for automated DS annotation based on UAV imagery and a transformer-based deep learning model called SugarViT. Hence, it is the second building block of the spread prediction workflow.

The most important data basis for this chapter is the individualized plant image catalog gained from Chapter 2. Thanks to the special imaging devices used in the field experiments, the images consist of multispectral UAV data with, in total, five channels. It is richer in its spectral information than common red, green, and blue channel (RGB) imagery, especially in those spectral regimes that are crucial for phytopathological processes. [25, 160] Several phenotyping studies utilizing UAVs and unmanned ground vehicles (UGVs) have already been done [10, 47, 63]. This chapter is about enhancing those studies by intelligent data-driven modeling and incorporation of knowledge.

Rehashed UAV image data has the potential to serve as training data even for large image-processing deep learning models. In recent years, the visual disease severity assessment with mainly convolutional neural network (CNN)-based models has made significant progress [133]. With breakthrough results regarding the application of transformer-based model architectures in diverse research areas [79], they are increasingly used for disease severity assessment as well. The origin of the transformer architecture lies in the field of natural language processing (NLP) and has led to great success in recent large language models such as the generative pretrained transformer (GPT) [66]. Its foundation, the attention mechanism (cf. Section 1.2) enables the model to connect and associate features over large semantic or sequential distances. This is beneficial not only for one-dimensional tasks such as NLP, since that context can be transferred to higher dimensional use cases like image processing. In this case, one is dealing with a ViT [31] model. Recent works like [54] and [44] use ViTs for plant disease localization and classification. However, common image processing deep learning architectures are mostly designed for three-channel RGB images and their image normalization strategies are designed accordingly. Thus, the normalization needs to be generalized to the multispectral case first. Moreover, the image data is augmented to further increase the effective data size.

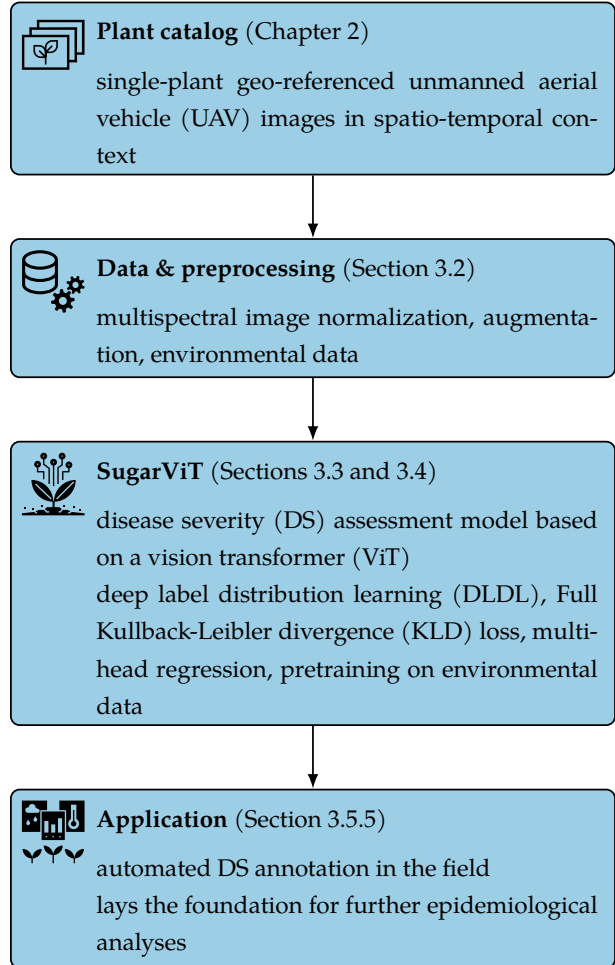


Figure 3.1: Conceptual overview of the disease severity prediction chapter.

Beyond the image data, one can further utilize the environmental data recorded with the in-field internet of things (IoT) sensors and connect them with weather station data. In the following, the transformer-based model is motivated and described in detail. It will use the largely available environmental data for pretraining before being finetuned on the expensive DS annotations. This is necessary because with the power of transformers, also a major drawback appears, namely their low data efficiency. Transformers need lots of data to train. This is why their success is currently mainly in application fields where large datasets are available, such as text data. However, this chapter shows that also on large-scale agricultural datasets enabled by UAVs, those models can be used for annotation tasks. SugarViT motivates a multi-objective approach as well as the use of a deep learning architecture based on a ViT. In contrast to [54] and [44], the DS prediction is identified as an ordinal classification and one can reinterpret the classification as a regression task by using the concept of deep label distribution learning (DLDL) introduced in [38]. Furthermore, the vanilla DLDL approach is optimized by an improved loss function that does not require dedicated hyperparameter tuning [C]. After training, SugarViT is able to predict the disease severity of individual plant images as a probability distribution, which improves training robustness and output interpretability. With its ViT backbone, it is able to estimate the DS more accurately than with convolution-based backbones of comparable complexity.

In the end, the automated DS annotation capabilities lay the foundation for further epidemiological analyses. To summarize the main components of this chapter, Figure 3.1 gives a schematic overview.

3.2 Data and Preprocessing

In this section, the underlying data for the model and the associated preprocessing methods are presented and clarified. A major challenge that comes with the application and, particularly, the training of vision-transformer-based architectures is the demand of large amounts of image data. In principle, the utilization of UAV imaging from crop fields has the potential to gain those large datasets. However, the conditions under which the images are taken can be very diverse, e.g., due to variable weather, lighting, and resolution. Additionally, device-specific properties can come into play when dealing with different camera models or calibration methods. In the context of plant phenotyping, it is particularly desirable to accumulate image data from multiple growing seasons, which implies that all the above-mentioned difficulties can play a role for an accumulation of large-scale datasets. Thus, in order to utilize as much potential from the data, a preprocessing pipeline is needed that is robust against as many confounding factors as possible.

3.2.1 Using Weather Data for Environmental Features

All experiment fields are equipped with weather sensors, allowing for hourly temperature and humidity measurements in the fields. One can use this data to infer some more environmental quantities, particularly focusing on two of them. Firstly, a basic yet widely used quantity in phytology that connects the local weather with the development stage of the crop is the cumulative growing degree days (GDDs). For hourly temperature data $T(d, h)$ for each day d and hour of day h , the GDD is the daily temperature mean between the base temperature T_{base} and maximum temperature T_{max} . Colder or warmer temperatures are clipped to this range. Hence, the so-called *thermal sum* is calculated by

$$\text{GDD}(d) = \frac{1}{24} \sum_{h=1}^{24} \min \left(\max (T(d, h), T_{\text{base}}), T_{\text{max}} \right) - T_{\text{base}}, \quad (3.1)$$

where the base and maximum temperatures T_{base} and T_{max} are plant-specific parameters. For sugar beet, it is empirically shown that $T_{\text{base}} = 1.1^\circ\text{C}$ and $T_{\text{max}} = 30^\circ\text{C}$ [56]. The cumulative quantity of GDDs beginning at the sowing date is, after all, a proxy of the plant's development. Secondly, a disease-specific quantity can be calculated. Simply put, the time between the infection of a plant with CLS and the ability of infecting other plants is called a *generation* or *incubation period*. The thermal sum of one incubation period for CLS in sugar beet is found to be 4963°C h with $T_{\text{base}} = 6.3^\circ\text{C}$ and $T_{\text{max}} = 32^\circ\text{C}$ [12]. For each hourly summand, there is an additional empirical correction coefficient based on the relative humidity: the hourly summand is multiplied by $\frac{7}{8}$ if the hourly relative humidity is less than 80 %. If it is at least 80 %, the summand is multiplied by $\frac{9}{8}$. [13, 60] Summation of the thermal sum and division by the incubation period yields a quotient that describes the potential number of incubation periods a hypothetically infected plant could have undergone, further referred to as the number of possible generations (NPG).

Thus, when environmental information is given, field- and recording-date-specific parameters that can serve as additional data to support the individual plant image data can be derived. Although GDD and NPG are not plant specific, they are, at least, specific for the recording date. Thus, one can annotate many plants with those labels at one stroke. These labels are, of course, not as informative as manually annotated labels, but they can serve for pretraining models. This is particularly interesting for an application of transformers, since they usually need lots of data: one can *pretrain* the model with the cheap labels and finetune on the expensive labels. Thereby, a possible lower availability of the expensive labels could be compensated and training speed is enhanced if the model backbone at the start of finetuning stage already “knows” low-level filters and the basic concept of the input data. The two different stages of pretraining and finetuning are represented as different learning paths in the model sketch in Figure 3.4.

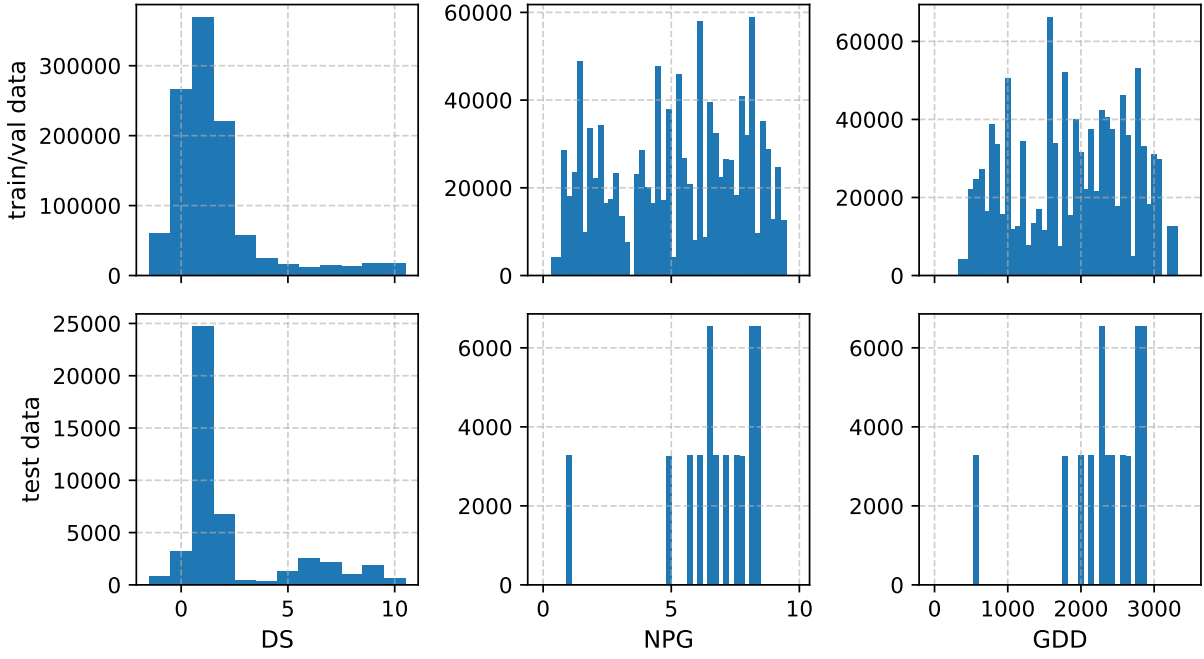


Figure 3.2: Histograms of available labels for disease severity (DS), number of possible generations (NPG), and growing degree day (GDD) separated by train/validation and test data.

3.2.2 Image Normalization

In the vast majority of machine learning tasks dealing with image processing, images are normalized to ensure interoperability and robustness against varying image recording conditions. Additionally, numerical issues in forward and backward pass to deep learning architectures lead to the usage of data values around zero. A naive yet common approach is a simple standardization of the image data by subtracting the channel-specific mean μ_c or a global mean μ and division by the standard deviation (std) σ_c or σ , respectively, for each image channel $\mathbf{C}$ in the image $\mathbf{I}$. The standardization can be done with precalculated (channel-wise) means and stds by using the information of the whole given dataset, with image-specific means and stds, or even with fixed, suggested values. In this chapter, it is assumed that the reflectance images dataset may be biased. Therefore, each image is standardized by only using its own information. Further, one distinguishes between channel-wise and a total standardization by using means and stds for each channel separately (μ_c, σ_c) and cross-channel (μ, σ) calculations, respectively. Thus, one yields channel-wise standardized images $\mathbf{I}_s^{\text{ch}}$ and total standardized images $\mathbf{I}_s^{\text{tot}}$ by

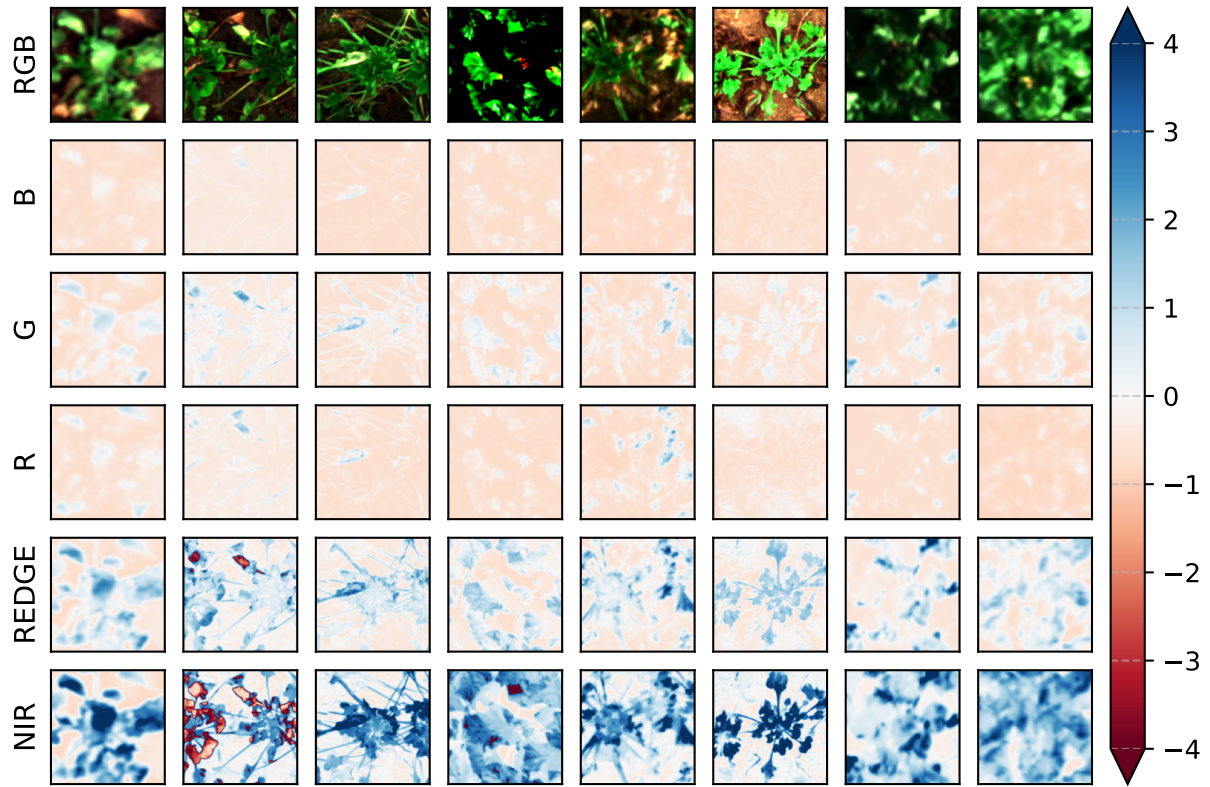
$$\mathbf{I}_s^{\text{ch}} = \left\{ \frac{\mathbf{C} - \mu_c}{\sigma_c} \right\}_{\mathbf{C} \in \mathbf{I}}, \quad (3.2)$$

$$\mathbf{I}_s^{\text{tot}} = \frac{\mathbf{I} - \mu}{\sigma}. \quad (3.3)$$

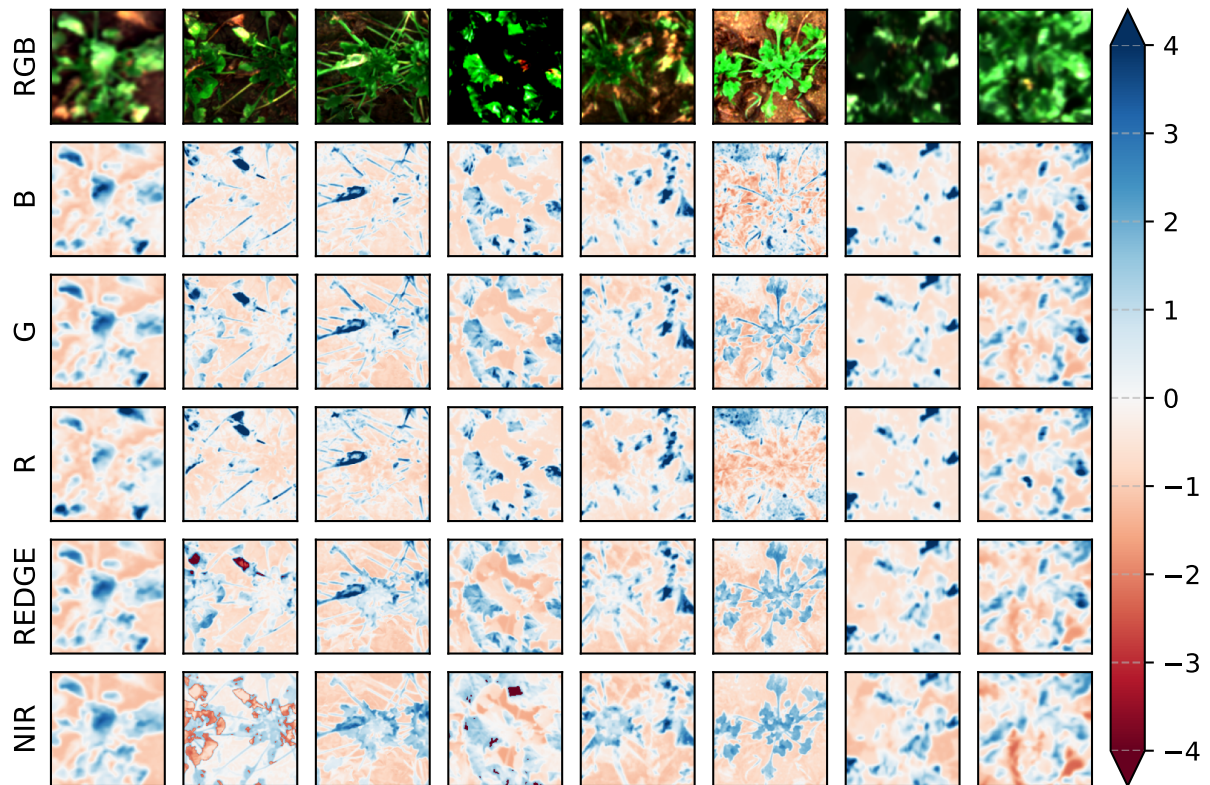
Figure 3.3 shows some example images separated into their channel components and normalized with those two standardization methods. It is clearly visible that the total standardization preserves the reflectance differences of the channels due to the emission spectrum of the plants and results in very distinct image channels. Thus, this standardization method emphasizes spectral characteristics of the image. In contrast, the channel-wise standardization results in rather similar image channels emphasizing local characteristics of the image in each channel.

A more sophisticated normalization method that, however, comes with more computational effort, makes use of the image histogram, i.e. the abundance of data values in the images. The histogram equalization (HE) is a contrast enhancement method that is broadly used in computer vision and image processing tasks for many application fields like medical imaging, as well as for signal processing like in speech recognition [110]. Simply put, with respect to images, particularly, the idea is to normalize the elementary pixel values by their abundance. Thus, the number of pixels in each bin, or range of contrast, is equalized. As a result, each image is forced to use the full range of possible contrast.

Generally, there are two basic methods—local and global. Unlike global methods, local methods additionally use the environment of the corresponding pixel for equalization. They are usually grouped under the term adaptive histogram equalization (AHE) where a prominent method is called contrast-limited adaptive histogram equalization (CLAHE) [113]. The adaptive methods are used for image processing tasks, where the pure contrast between neighboring objects is important, like in medical applications for tomography images [4]. In this workflow, the HE method is applied and a channel-wise and cross-channel variant analogous to the standardization is introduced. For the histogram-based methods, a lower and upper limit has to be predefined to which the values are scaled. Here, the values are chosen to be in the range $[-1, 1]$.



(a) Total standardization.



(b) Channel-wise standardization.

Figure 3.3: Example images shown by its separate channel components and processed with total and channel-wise standardization, respectively. The first row shows the RGB representation. The other rows show the separate channels with increasing wavelength from top to bottom.

3.2.3 Data Augmentation

A common approach to artificially increase the dataset size is data augmentation. For image data, the principle is to sample similar images around “real” data instances by various techniques like flip, rotation, color and brightness jitter, etc. Obviously, different use cases allow for different augmentation methods. For instance, in case of medical imaging tasks, one is mostly bound to the image orientation. In case of street scene images for autonomous driving applications, a vertical flip, i.e. put the image upside down, does not make any sense. However, both cases eventually allow for changes in brightness and/or contrast. In the case of plant images, fortunately all degrees of freedom regarding flips and rotations are allowed. Thus, each image is flipped randomly with a probability of 25 % and rotated by a random angle. Brightness and contrast jitters are not necessary, since the normalization methods neutralize them. Additionally, one can exploit this principle in model inference mode by evaluating the images in different rotations and average the predictions. In order to be robust against different ground sampling distances and accompanied resolution changes, a Gaussian blur augmentation is used. With a probability of 10 %, an image is blurred at a strength of 3 – 8 px. Another optional augmentation, is a random channel dropout. With a probability of 25 %, the information of up to 3 channels is dropped. Although it is quite unlikely that in the application, single channels will be dropped, it is interesting to train models being robust against missing information in order to see, how important each image channel is for the prediction of the target quantity. By channel dropout, the model is forced to connect information among all channels rather than focus on single channels.

Now that the data (pre-)processing is explained, the model architecture is motivated and described in detail in the following sections.

3.3 Deep Label Distribution Learning

If classification problems can be formulated within an ordinal scale, the transfer into a regression task might be a good choice. However, if the classification is very granular, the collection of data with precise labels can be challenging. Rather than learning distinct, unique labels, the paradigm of label distribution learning (LDL) [40] was proposed. It stabilizes the model training of labels by modeling their ambiguity. It is used for tasks like facial age estimation [42] or head pose estimation [41]. In combination with deep neural networks (NNs), the paradigm is referred to as deep label distribution learning (DLDL) [38]. In DLDL, the output of a deep NN mimics the label distribution by a series of neurons that learn a discrete representation of the probability density function. This representation is commonly known as the probability mass function (PMF) and labels have the form of a probability distribution. The obvious difference in contrast

to a pure regression is that the network output is not only based on a single neuron. Moreover, the difference to a pure classification is that, in contrast to one- or multi-hot-labels, also neighboring neurons are triggered which stabilizes regions, where fewer data is available. Two additional advantages, especially for the use case in this chapter, are, firstly, that the uncertainty of labels can be easily modeled. The DS annotation is based on individual human experts' judgment. Often, different plants are annotated by different experts, which causes uncertain classifications. Secondly, the model output becomes more transparent, since one can observe how confident the model is in its prediction by comparing the shapes of true and predicted label distribution. Thus, DLDL, once more, is an ideal way to model these annotations.

3.3.1 Full Kullback-Leibler Divergence Loss

The DLDL approach proposed in [38] utilizes a ℓ_1 loss for the expectation value and a KLD [65] loss for the label distribution. However, ℓ_1 and KLD loss originate from different statistical concepts and, therefore, have scales that are, per se, not comparable. In most cases, a weighting parameter has to be introduced, resulting in an artificial hyperparameter of the model. However, the novel approach circumvents the problem by reformulating the ℓ_1 loss as a KLD loss. Additionally, the training is further accelerated by introducing a "smoothness" regularization to the label distribution. The regularization is also formulated as a KLD loss, not needing any hyperparameter, either. Furthermore, the gained scale invariance not only makes the components comparable, but also enables the cross-comparability between different labels. This is especially interesting in this very use case, since it aims a joint regression of diverse phenological parameters, probably having different domains. Thus, the essentials of the novel three-part loss function are explained in the following. The full elaboration has been done in [C].

Label Distribution Loss Let $\mathbb{P}(y \mid \mathbf{x})$ be the true label distribution for a given input image $\mathbf{x}$. Then, the label distribution loss $\mathcal{L}_{ld}$ is the discrete Kullback-Leibler divergence between the true and predicted label distribution,

$$\mathcal{L}_{ld} := \text{KLD}(\mathbb{P} \parallel \hat{\mathbb{P}}) = \sum_y \mathbb{P}(y \mid \mathbf{x}) \log \frac{\mathbb{P}(y \mid \mathbf{x})}{\hat{\mathbb{P}}(y \mid \mathbf{x})}, \quad (3.4)$$

where the hat denotes the prediction. This definition follows the label distribution loss given in [38].

Expectation Value Loss Unlike in [38], the expectation value loss is formulated as a KLD of truth and prediction as if both of them were normal distributions $\mathcal{N}(\cdot \mid \mu, \sigma^2)$

with expectation value μ and variance σ^2 . For the model predictions, $\hat{\mu}$ and $\hat{\sigma}^2$ can be calculated via

$$\hat{\mu} = \mathbb{E}[\hat{\mathbb{P}}] = \sum_y y \hat{\mathbb{P}}(y | \mathbf{x}), \quad \hat{\sigma}^2 = \text{Var}[\hat{\mathbb{P}}] = \sum_y (y - \hat{\mu})^2 \hat{\mathbb{P}}(y | \mathbf{x}). \quad (3.5)$$

Thus, the expectation value loss is

$$\begin{aligned} \mathcal{L}_{exp} &:= \text{KLD}(\mathcal{N}(\cdot | \mu, \sigma^2) \| \mathcal{N}(\cdot | \hat{\mu}, \hat{\sigma}^2)) \\ &= \sum_y \mathbb{P}(y | \mathbf{x}) \log \mathbb{P}(y | \mathbf{x}) - \sum_y \mathbb{P}(y | \mathbf{x}) \log \hat{\mathbb{P}}(y | \mathbf{x}) \\ &= - \sum_y \mathbb{P}(y | \mathbf{x}) \left[\frac{1}{2} \log(2\pi\sigma^2) + \frac{(y - \mu)^2}{2\sigma^2} \right] + \sum_y \mathbb{P}(y | \mathbf{x}) \left[\frac{1}{2} \log(2\pi\hat{\sigma}^2) + \frac{(y - \hat{\mu})^2}{2\hat{\sigma}^2} \right] \\ &= - \frac{1}{2} \log(2\pi\sigma^2) \underbrace{\sum_y \mathbb{P}(y | \mathbf{x})}_{=1} - \frac{1}{2\sigma^2} \underbrace{\sum_y (y - \mu)^2 \mathbb{P}(y | \mathbf{x})}_{=\sigma^2} \\ &\quad + \frac{1}{2} \log(2\pi\hat{\sigma}^2) \underbrace{\sum_y \mathbb{P}(y | \mathbf{x})}_{=1} + \frac{1}{2\hat{\sigma}^2} \underbrace{\sum_y (y - \hat{\mu})^2 \mathbb{P}(y | \mathbf{x})}_{=\mu^2 + \sigma^2} \\ &= \log \frac{\hat{\sigma}}{\sigma} - \frac{1}{2} + \frac{\sum_y y^2 \mathbb{P}(y | \mathbf{x}) - 2\hat{\mu} \sum_y y \mathbb{P}(y | \mathbf{x}) + \hat{\mu}^2}{2\hat{\sigma}^2} \\ &= \log \frac{\hat{\sigma}}{\sigma} - \frac{1}{2} + \frac{\sigma^2 + (\hat{\mu} - \mu)^2}{2\hat{\sigma}^2}. \end{aligned} \quad (3.6)$$

Smoothness Regularization Loss In order to accelerate the training process, especially in early stages, the predicted label distribution is forced to be smooth by a KLD regularization term. The idea is to shift the predicted distribution by one position $\hat{\mathbb{P}}^s$ and calculate a symmetric discrete KLD, i.e. both shift directions are averaged. Thus,

$$\begin{aligned} \mathcal{L}_{smooth} &:= \frac{1}{2} [\text{KLD}(\hat{\mathbb{P}} \| \hat{\mathbb{P}}^s) + \text{KLD}(\hat{\mathbb{P}}^s \| \hat{\mathbb{P}})] \\ &= \frac{1}{2} \left[\sum_y \hat{\mathbb{P}}(y | \mathbf{x}) \log \hat{\mathbb{P}}(y | \mathbf{x}) - \sum_y \hat{\mathbb{P}}(y | \mathbf{x}) \log \hat{\mathbb{P}}^s(y | \mathbf{x}) \right. \\ &\quad \left. + \sum_y \hat{\mathbb{P}}^s(y | \mathbf{x}) \log \hat{\mathbb{P}}^s(y | \mathbf{x}) - \sum_y \hat{\mathbb{P}}^s(y | \mathbf{x}) \log \hat{\mathbb{P}}(y | \mathbf{x}) \right] \\ &= \frac{1}{2} \left[\sum_y (\hat{\mathbb{P}}(y | \mathbf{x}) - \hat{\mathbb{P}}^s(y | \mathbf{x})) \log \hat{\mathbb{P}}(y | \mathbf{x}) - (\hat{\mathbb{P}}(y | \mathbf{x}) - \hat{\mathbb{P}}^s(y | \mathbf{x})) \log \hat{\mathbb{P}}^s(y | \mathbf{x}) \right] \\ &= \frac{1}{2} \sum_y (\hat{\mathbb{P}}(y | \mathbf{x}) - \hat{\mathbb{P}}^s(y | \mathbf{x})) \log \frac{\hat{\mathbb{P}}(y | \mathbf{x})}{\hat{\mathbb{P}}^s(y | \mathbf{x})}. \end{aligned} \quad (3.7)$$

Finally, the loss components are combined into a sum that gives the loss as

$$\mathcal{L} := \mathcal{L}_{ld} + \mathcal{L}_{exp} + \mathcal{L}_{smooth} . \quad (3.8)$$

3.3.2 Multi-Head Regression

If multiple sources of labels are available, it may be considerable to perform the regression with multiple labels jointly. Each regression problem is then realized by its own so-called “regression head”, i.e., a sub-model that is trained to transform the feature representation from the backbone into the respective label space of interest. Especially for large backbone models, this has the advantage that only one backbone is needed for multiple purposes, which reduces the total model size. In the following, this concept is called “Multi-Head Regression”.

3.4 SugarViT Architecture

In this chapter, the architecture of the proposed model is described. Figure 3.4 shows all the building blocks of the proposed model—further called *SugarViT*—and the three main building blocks of SugarViT and its design motivations are described in the following.

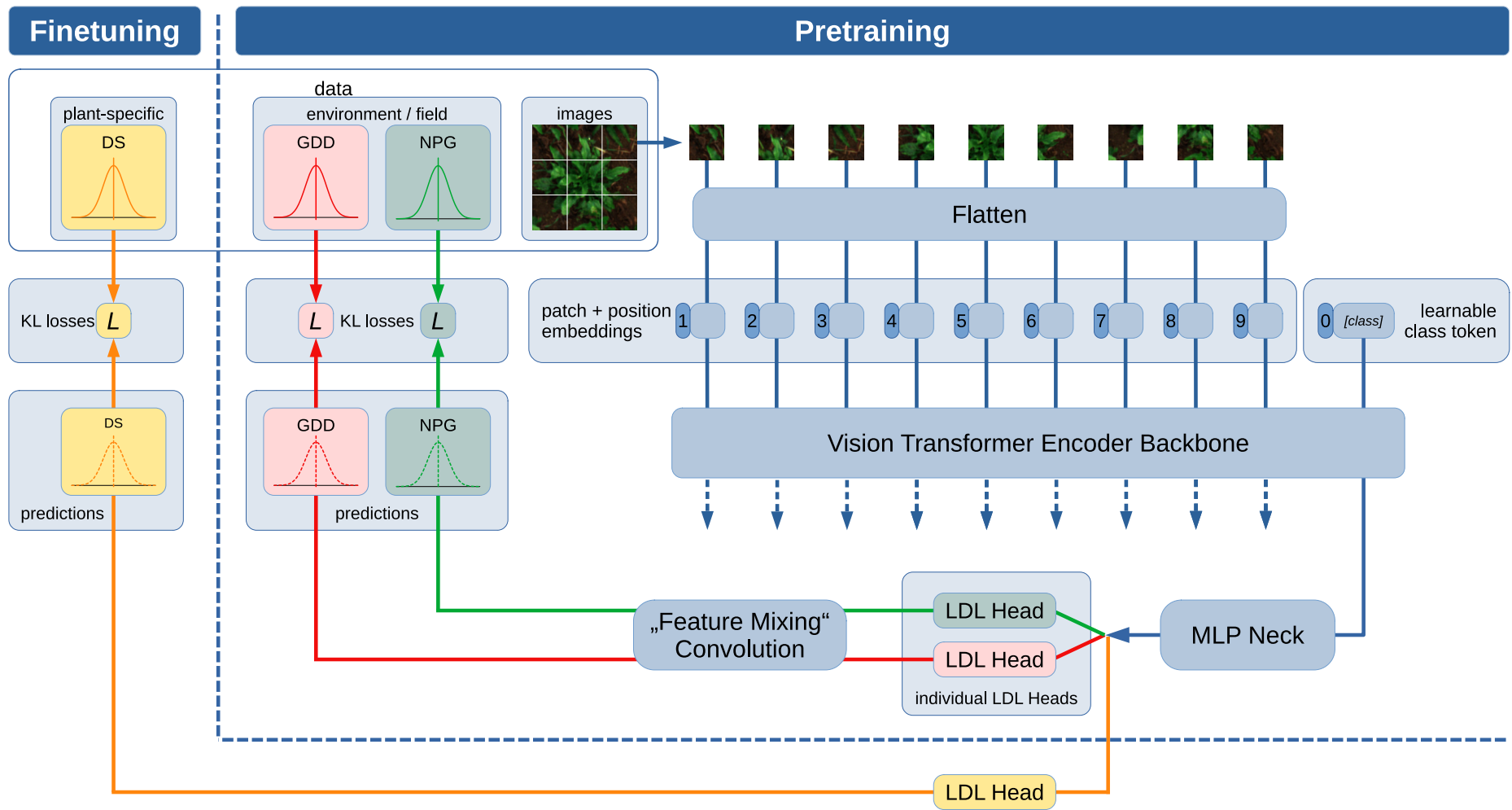


Figure 3.4: Sketch of SugarViT, the proposed Multi deep label distribution learning (Multi-DL DL) network with a vision transformer (ViT) backbone. The label distribution learning (LDL) heads are trained with separate optimizers and loss functions. The ViT and multi-layer perceptron (MLP) part are the joint basis and are trained in each backward pass of the LDL heads. As output of the ViT, the last hidden state of the learnable class token is used. Furthermore, the DS estimation use case is shown by having multispectral plant image data and two training stages. The pretraining is done on the environmental, field-related quantities growing degree day (GDD) and number of possible generations (NPG). The target label disease severity (DS) is trained in the subsequent finetuning stage. In principle, the model can be generalized to more labels in each training stage by adding more LDL heads.

3.4.1 Vision Transformer Backbone

In recent years, transformer architectures have been successfully utilized for diverse deep learning tasks. The underlying attention mechanism [144] (cf. Section 1.2) is able to relate patterns and semantics in sequential data very flexibly and across large structural distances. Especially in the field of NLP, transformer-based models show great success. [66]. In NLP, transformers learn structures in sequential data like text or sentences by processing its basic building blocks, commonly known as “tokens”. To use this principle also for classification tasks on image data, the vision transformer (ViT) model has been proposed. [31] The main principle is to divide an input image into flattened tiles that are processed by several multi-head attention layers. An additional learnable “class token” is added, whose processed output is passed through a classification head. Figure 3.4 also shows the mentioned building blocks. By comprising many attention blocks and hidden layers, (vision) transformer architectures are complex and need large amounts of data. Thus, they are usually pretrained on large-scale datasets like ImageNet [126] for most of the image processing tasks. For the use case this chapter is about, multispectral 5-channel images rather than red, green, and blue channel (RGB) images are processed. Therefore, one cannot use ImageNet-pretrained architectures per se. However, the plant image dataset used in this dissertation is large enough to train an architecture with a ViT backbone from scratch, as further sections will show. The goal of the learning process is that the ViT backbone is trained to be an expert in understanding the image as a whole and extract remarkable traits to “encode” the image information into a rich feature space.

3.4.2 MLP Neck

In the original ViT model, a multi-layer perceptron (MLP) is used as a classification head. Since a multi-head regression model should be built (cf. Section 3.3.2), an MLP is used as an intermediate layer between the ViT backbone and the regression heads. If certain labels have something in common or share some information—i.e. latent label correlations—this “neck” sub-model between backbone and heads is trained to learn those latent correlations. The principle can be exploited in this use case by introducing a simple “cheap” feature that is easy to measure and has a more or less obvious correlation to the DS. For instance, one could choose the interval between the image recording date and the date where canopy closure can be observed in the corresponding field experiment. Canopy closure means that neighboring plants touch each other, resulting in a closed field vegetation. In the following, this feature is called “days after canopy closure (DAC)”. Obviously, this feature has negative values before canopy closure is reached. Alternatively, one could also think about including the days after sowing. The DAC are

expected to guide the model to the correct DS by having some correlation with it, e.g., young plants (low DAC) are probably less severely infected, whereas older plants (high DAC) are probably rather severely infected. In addition, the infection probability increases when the plants are in contact. All in all, the MLP neck is the part of the model, where expert knowledge and known correlations can be integrated. Please note here that the experiments follow another approach to integrate associated knowledge in the model to predict the DS. Nevertheless, the above approach can be a valid one as well.

3.4.3 LDL Heads

For each label, the output of the MLP neck is processed by a separate, so-called, LDL head. It consists of individual fully-connected layers (FCLs) after the MLP neck. The idea behind the individual networks is to enable the model to learn label-individual transformations from the cross-label output of the MLP neck. Thus, the LDL heads are trained to be experts in their label domain and be able to transform the feature space to their respective label space. When passed through these layers, the features are mixed with a component called *Feature Mixing*.

3.4.4 Feature Mixing

Different labels can contain different amounts of information or be differently difficult to predict. Moreover, they could be (anti-)correlated or complement each other. Thus, the model is assumed to profit from a layer that can relate or mix high-level features learned in the previous layers with each other. In a so-called Feature Mixing component, the output layers of all individual LDL Heads are combined linearly. This enables the model to scale and mix information of other labels into the actual labels. The mixing coefficients can be learned during training and are initialized to the unit matrix, i.e., at training start, only the respective label is used. A final FCL for each label maps the mixed features into the corresponding label distribution space. The size of the FCL is determined by the number of discretization or quantization steps which can be different for each label. A softmax activation defined as

$$\text{softmax}(y_i) = \frac{\exp(y_i)}{\sum_i \exp(y_i)} \quad (3.9)$$

ensures the outputs of the FCLs sum to 1 each. Thus, the FCL approximates the label distribution in the form of a PMF. On this PMF, one can then evaluate the KLD loss functions and, finally, train with the ground truth label distributions.

3.5 Results

In this section, the performed experiments are introduced. At first, it is tested whether the histogram equalization preprocessing step for the images is really beneficial for the model performance.

In the next experiment, one makes use of “cheap” data, i.e., weather data as mentioned in Section 1.4.2. In this very case, the available IoT sensors in the field are measuring basic weather parameters. This data is available for a whole field, thus, many single plant images. With the single images and those cheap labels, one can perform a pretraining of the backbone. This is another approach to combine cheap and expensive data than the one mentioned in Section 3.4.2. However, the major advantage of this approach is that in the final model, only the label of interest is used, which results in a slightly lighter model and decreases inference time. After pretraining, a model training with DS labels is performed, resulting in the SugarViT model. Example predictions of a trained SugarViT model are shown in Figure 3.5.

In a last experiment, it is finally investigated if the pretraining also improves the performance of SugarViT by comparing the finetuned SugarViT with one trained only on the DS labels. Furthermore, a non-pretrained SugarViT model is compared to one that is only trained on RGB-band information in order to see whether the beyond-optical spectral information is important.

Before performing the actual experiments, the variances for the DS label distributions must be set, since there is no individual information for each data point, or image. In this chapter, the DS label distributions are modeled by normal distributions $\mathcal{N}(\cdot \mid \mu, \sigma^2)$ with the experts’ labels as expectation values μ and a variance σ^2 that is based on an assumed standard error. Exemplary, let $\sigma_{DS} = 0.6$ be the “human estimation” error. Please note that this is no empirically found error but rather an educated guess. In the scope of this chapter, only one human estimate per plant and per acquisition date is available. If multiple experts are involved, one could determine a more realistic error value or even a more realistic probability density function (PDF). For now, the estimate of a normal distribution with fixed std should be seen as an exemplary choice. Being a usual choice of optimization algorithm for transformer-based models, the AdamW [81] optimizer is used in all experiments.

In all experiments, the training and validation data (cf. Table 1.3) is randomly split with initialization seeds assuring reproducibility. For the given dataset, the plants have mostly low DS scores in images of the early plant growth, leading to label imbalance, as seen in the histograms in Figure 3.2. Looking at Table 1.3, the sizes of the datasets are quite different. In order to minimize the bias and to prevent the model from focusing on few labels and datasets, a weighted sampling of the data is performed. The weight of each image is the inverse of the total abundance of its DS label times the size of the

respective dataset. Thus, in each training batch, the distribution of datasets and labels is uniformly distributed on average.

Finally, a validation metric needs to be introduced. Since the model outputs distributions, metrics like root mean squared error (RMSE) or mean absolute error (MAE) are not appropriate since they do not give information about the overall distribution. Alternatively, the mean overlap between the predicted and true label distribution is used. Since the distributions are PMFs, the calculation of the mean distribution overlap (MDO) for a batch of N instances is

$$\text{MDO} := \frac{1}{N} \sum_{i=1}^N \sum_y \left\{ \min \left(\mathbb{P}(y \mid \mathbf{x}_i), \hat{\mathbb{P}}(y \mid \mathbf{x}_i) \right) \right\}_y. \quad (3.10)$$

The MDO takes values between 0 and 1 where 1 indicates perfect overlap. For the validation, the same weighted sampling as in the training stage is used to validate on pseudo-uniform distributed labels. Thus, one respects the prediction quality for each label equally and independently of the total label abundance in the dataset.

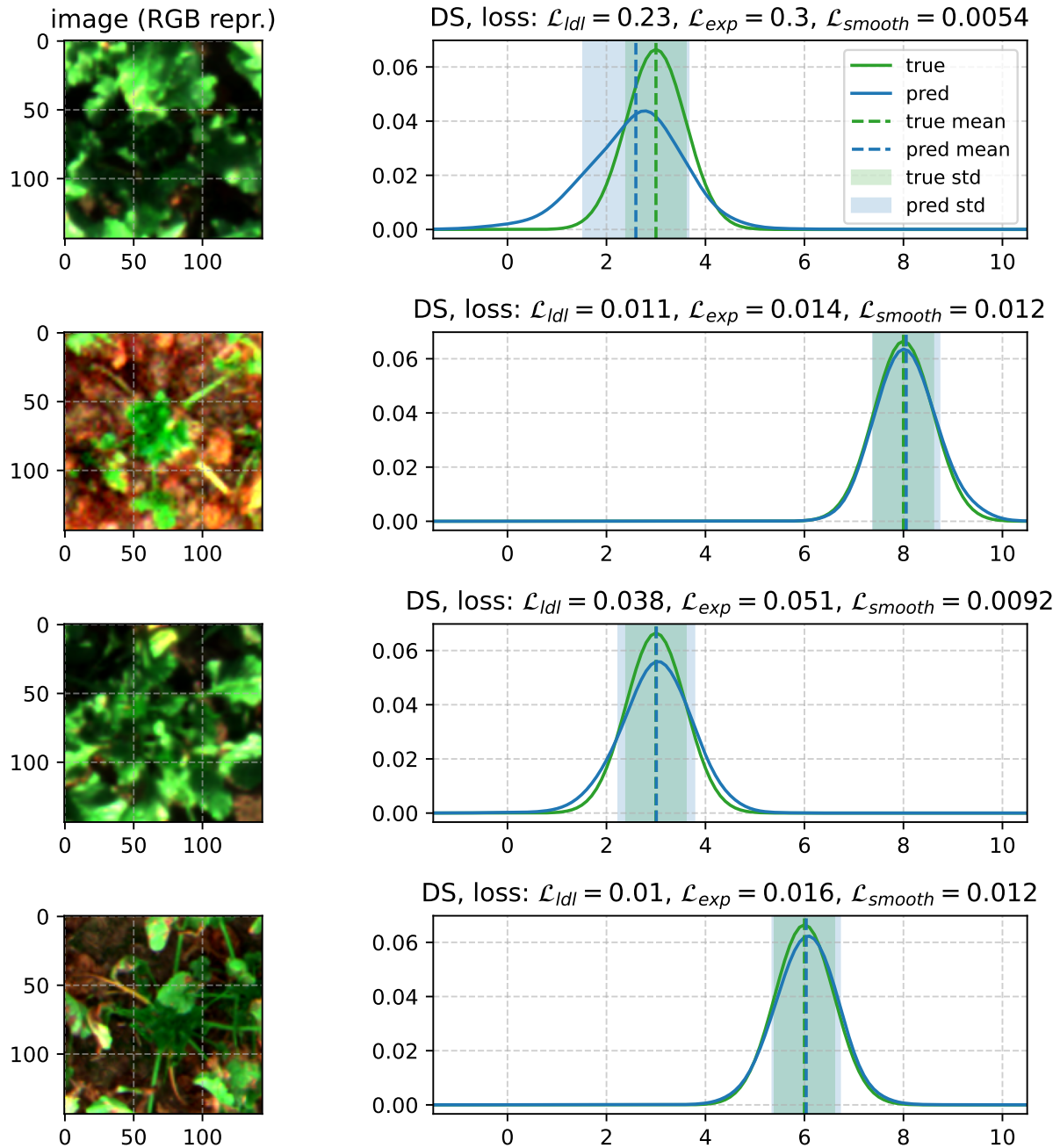


Figure 3.5: Output of SugarViT. The disease severity (DS) labels are learned as label distributions (green curves). SugarViT outputs again probability distributions (blue curve). The prediction in the end is the expectation value of the output distributions (dashed lines).

3.5.1 Standardization vs. Histogram Equalization

Before the training of SugarViT is performed, it is evaluated how the histogram equalization improves the model performance in favor of a “simpler” standardization preprocessing method. To make the model potentially more universal, no prior knowledge of the full dataset is assumed. Thus, the normalization is only based on the information of a single image, as already mentioned in Section 3.2.2. The following experiment compares the standardization and the histogram equalization method, respectively, with channel-wise and cross-channel calculation. Each method performs 10 runs with different initialization seeds with the model configuration given in Table 3.1. To speed up the training, only the arbitrarily chosen datasets D01a, D01b, and D02 (cf. Table 1.3) are used as a subset. Those are randomly split into a training and validation subset with a validation ratio of 20 %. This ratio is a good compromise between having a large training set and a sufficiently large validation set. Furthermore, the impact of using image augmentation is evaluated by performing all these experiments with and without using augmentation in the training stage.

The results of a training with 80 epochs are presented in Figure 3.6. As expected, the total normalization methods perform better than channel-wise normalization. This makes sense, because for DS prediction, an important feature is the difference in reflectance of spectral bands, thus, the difference in values across channels. When normalizing the image totally by calculation of cross-channel histograms or mean and std, respectively, this information is preserved, while in the channel-wise normalization it is lost. Nevertheless, channel-wise normalization is more robust against calibration errors of the sensor.

ViT backbone	
input size	$5 \times 144 \times 144$ px
patch size	12 px
hidden size	512
# hidden layers	4
# attention heads	4
intermediate size	512
activation hidden layers	GELU
dropout hidden layers	0.02
attention dropout	0.02
MLP neck	
layer size	512
layers	3
activation	ReLU
normalization	LN
dropout	0.2
LDL heads	
individual MLP layers	2
individual MLP layer size	256
activation	ReLU
dropout	0.8
DS quantization steps	131
DS regression limits	$[-2.0, 11.0]$
DS label distribution std	0.6
Optimizer	
algorithm	AdamW
initial learning rate	10^{-3}
weight decay	0.1

Table 3.1: Model configuration for comparison between standardization and histogram equalization.

Another remarkable observation is that the standardization method is not only computationally more efficient than histogram equalization, but also performs better. Thus, the total (cross-channel) standardization method is found to be the best performing normalization method, and is thus used for the SugarViT model. Additionally, the results reveal that the image augmentation is beneficial for the training process. The effect is low but visible especially for total standardization and channel-wise histogram equalization. Thus, all training stages will use image augmentation for further experiments.

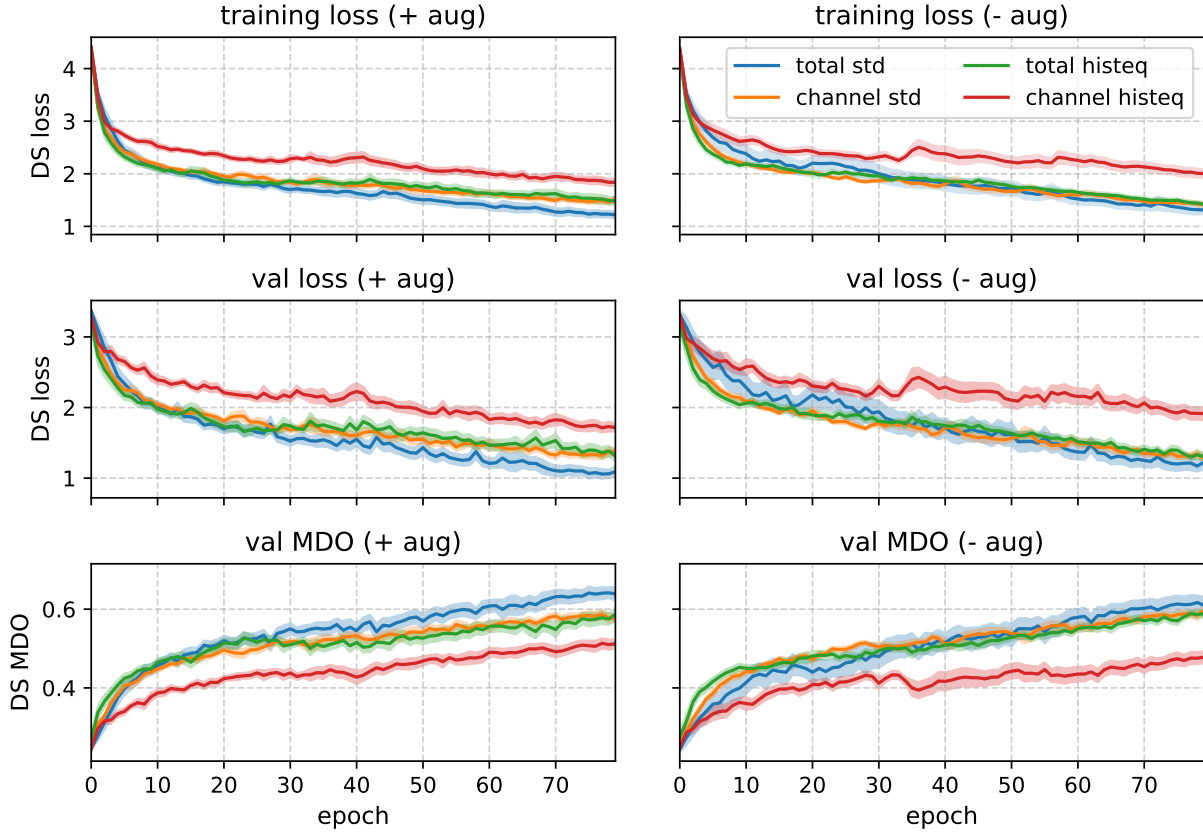


Figure 3.6: Results of the standardization vs. histogram equalization experiment. For both methods, total and channel-wise variants are shown. For each experiment, 10 runs with different seeds are performed. The bold lines describe the means, whereas the bands are the standard errors of the respective mean. The left plots show the results using augmentation during training and the right ones without augmentation. For each variant, training loss (top), validation loss (center), and validation MDOs (bottom) are plotted.

3.5.2 SugarViT Pretraining

SugarViT is pretrained on the environmental data labels. This should prepare the model for the plant images by learning low-level features of the plant images. The configuration of SugarViT for both pretraining and finetuning stage is listed in Table 3.2. Please note that most of the hyperparameters are chosen by educated guesses and are not optimized. A detailed hyperparameter search experiment would have exceeded the scope of this

chapter. Since most of the experiments demonstrated here are comparisons, the chosen hyperparameters are sufficient to show the model’s capabilities.

The results for training and validation loss, as well as the validation MDO are shown in Figure 3.7. As seen in the training loss component plot, the full KLD loss function is indeed invariant under the label scale, as described in Section 3.3.1. Without any scaling parameter, both GDD and NPG labels have a comparable loss, although the scales are very different. One sees a convergence of the validation MDO at ca. 90 % after roughly 40 training epochs for both GDD and NPG labels. As the best model, the one with the best validation MDO is taken and used for the further steps. The results for the training variants with channel dropout and RGB-only information are very similar as shown in Figure 3.8. Moreover, Figure 3.9 shows the training loss components for both variants. Again, the results are very similar to the full 5-channel training. Thus, one can conclude that the pretraining is robust against missing spectral information during training. Whether the pretraining was beneficial for the subsequent finetuning regarding convergence speed and prediction quality, is shown in the following section.

ViT backbone	
input size	$5 \times 144 \times 144$ px
patch size	12 px
hidden size	1024
# hidden layers	8
# attention heads	4
intermediate size	1024
activation hidden layers	GELU
dropout hidden layers	0.02
attention dropout	0.02
MLP neck	
layer size	512
layers	3
activation	ReLU
normalization	LN
dropout	0.2
LDL heads	
individual MLP layers	2
individual MLP layer size	256
activation	ReLU
normalization	LN
dropout	0.8
NPG quantization steps	100
NPG regression limits	$[-0.5, 11.5]$
NPG label distribution std	0.3
GDD quantization steps	100
GDD regression limits	$[-5^{\circ}\text{C d}, 3500^{\circ}\text{C d}]$
GDD label distribution std	100°C d
DS quantization steps	131
DS regression limits	$[-2.0, 11.0]$
DS label distribution std	0.6
Optimizer	
algorithm	AdamW
initial learning rate	5×10^{-4}
weight decay	0.1
Learning rate scheduler	
strategy	cyclic learning rate (linear)
interval	step
maximum lr	1×10^{-3}
step size (up and down)	500
mode	exponential range
γ	0.9999

Table 3.2: SugarViT model configuration for pretraining and finetuning. For pretraining, the labels number of possible generations (NPG) and growing degree day (GDD) are used. In finetuning stage, the final label of interest, disease severity (DS), is trained.

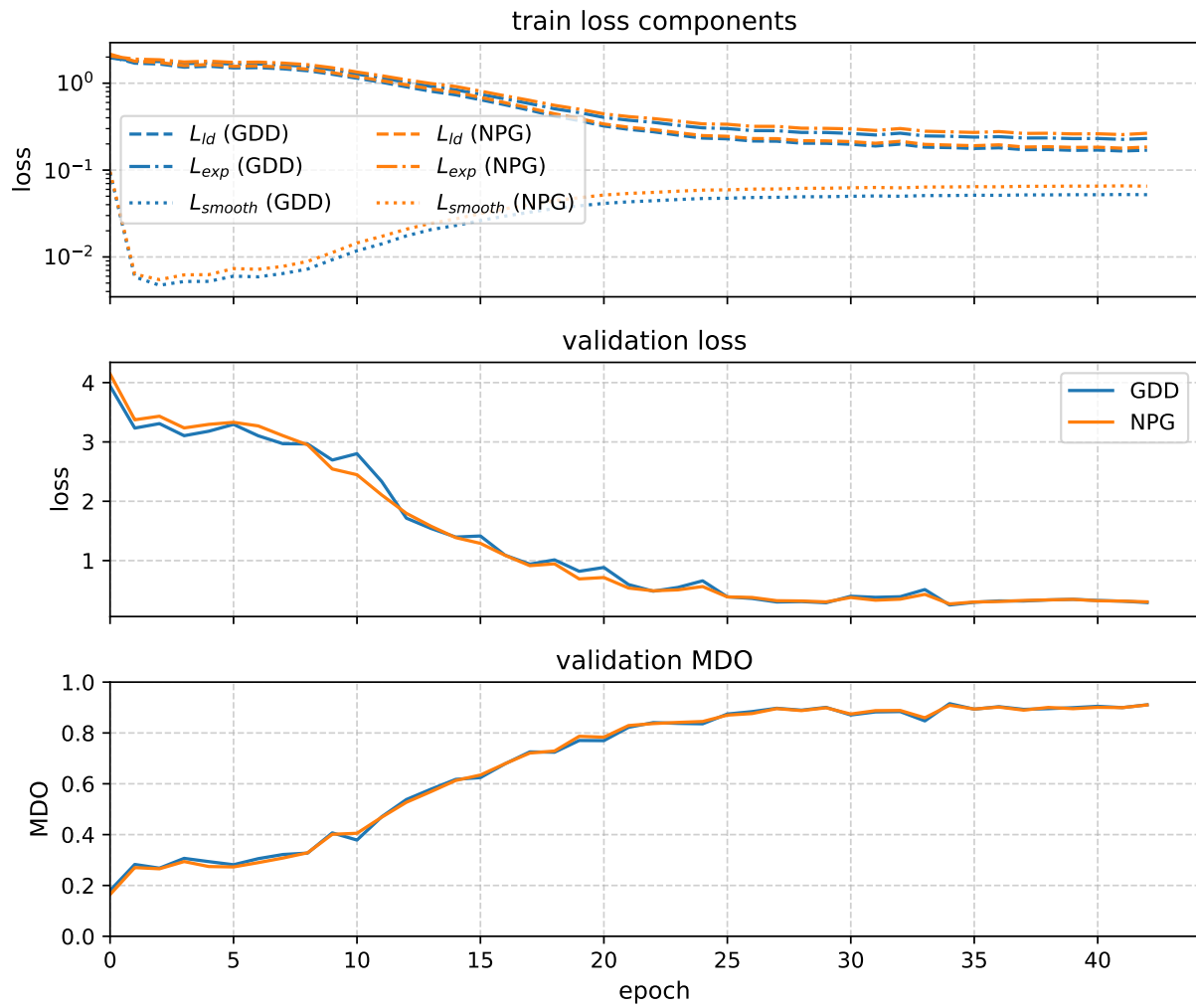


Figure 3.7: Results of the SugarViT pretraining. The top plot shows training loss by epoch separated into the 3 loss terms (cf. Equation (3.8)) for GDD (blue) and NPG labels (orange). The two plots below show metrics, namely validation loss (center) and validation MDO (bottom), against the training epoch for both labels.

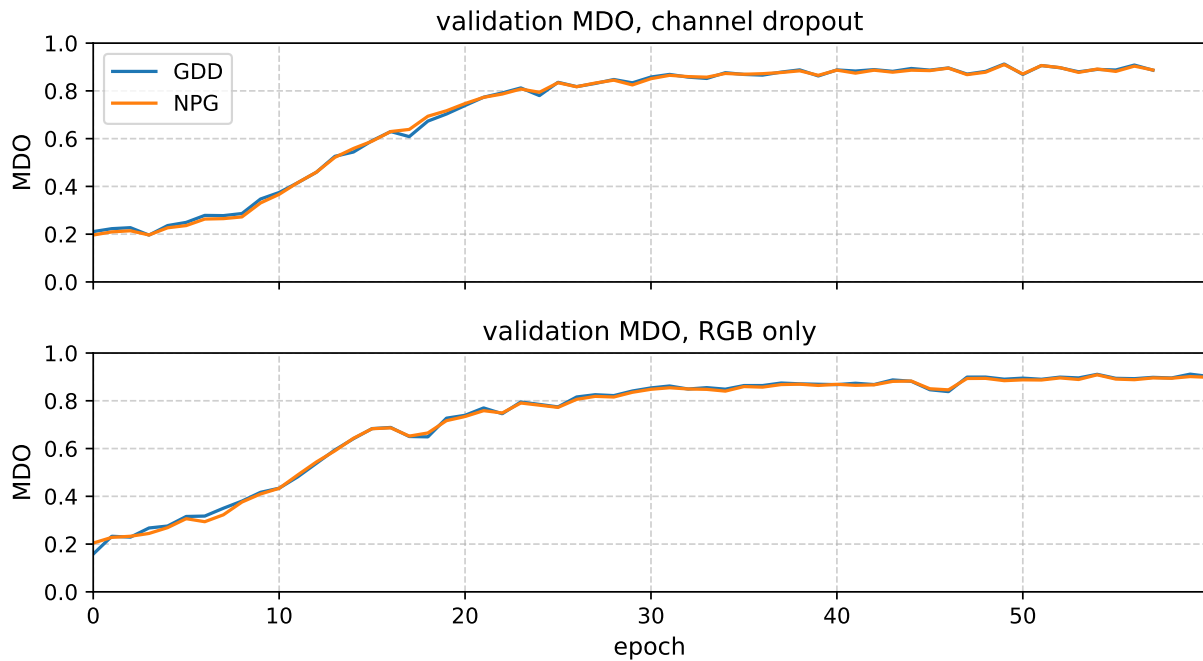


Figure 3.8: Validation mean distribution overlap (MDO) by training epoch of the SugarViT pretraining for the channel dropout (top) and the RGB-only (bottom) variants.

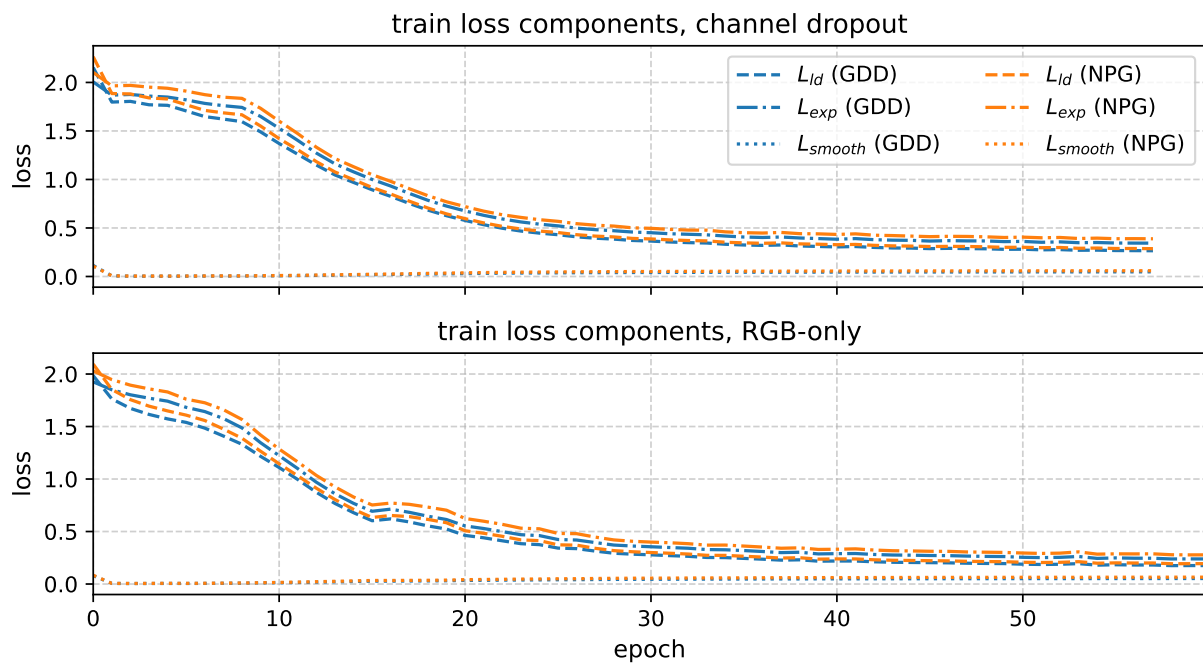


Figure 3.9: Training loss components by training epoch of the SugarViT DS training for the channel dropout (top) and the RGB-only (bottom) variants.

3.5.3 Comparison Experiments

In this section, two comparison experiments are performed to justify the choices of model pretraining and using a ViT backbone.

3.5.3.1 Backbone Network

First, let us take a look at the backbone network that gives SugarViT its name—the vision transformer (ViT). In order to see its benefits, it is compared to another backbone network that has a comparable complexity regarding the number of parameters. The ViT backbone has about 51.3×10^6 parameters. A ResNet-152 [52] being similarly complex with 60.2×10^6 parameters and a VGG-19 [134] with 45.7×10^6 parameters serve as baseline backbones. To preserve compatibility with the 5-channel input images, the number of input channels in the first layer has to be changed. All other model parts are untouched, and a full training from scratch with the same hyperparameters as for SugarViT is performed. As Figure 3.10 shows, the model with ResNet-152 backbone trains substantially faster. However, when it comes to the validation metrics MDO and MAE, it underperforms SugarViT regarding the best model metrics. Additionally, the ResNet-152 model seems to be less stable in the metrics due to a volatile behavior beyond roughly 40 epochs. The VGG-19 shows a more stable behavior and a similarly fast training loss convergence. The plateau in the beginning is due to the apparently too high learning rate. Since the same training procedure is performed for all models, the learning rate scheduler is not adapted to the VGG-19 model. Once the learning rate is low enough, the VGG-19 model starts the actual training. However, it underperforms SugarViT with about the same benchmarks as the ResNet-152 model.

The findings support statements regarding transformer-based networks demanding more data and patience in training. However, once the data amount is sufficient, they are indeed able to outperform convolution-based networks even with lower network complexity like—in this case—with about 15 % fewer parameters in comparison with ResNet-152. Nevertheless, the fast training behavior of ResNet-152 and VGG-19 suggests that using them for use cases with limited data and resource availability is still a good choice.

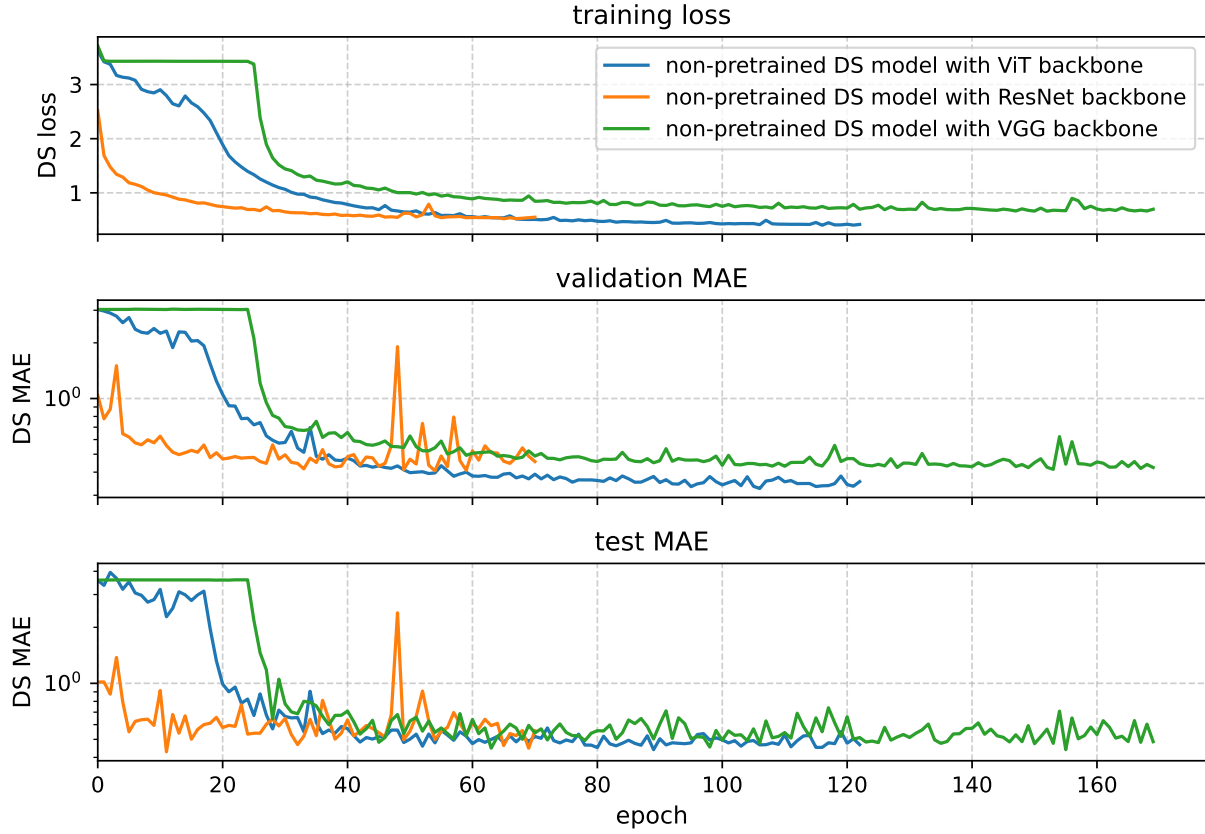


Figure 3.10: Comparison of backbone networks. Non-pretrained models for prediction of DS are trained using different backbones. The blue lines show the SugarViT model with ViT backbone, whereas the orange lines show the same model but with a ResNet-152 backbone. Green lines show it with a VGG-19 backbone. Training loss (top), validation MDO (center) and validation MAE (bottom), against the training epoch are presented.

3.5.3.2 Pretraining

Now, the training and validation metrics of the pretrained SugarViT compared to a non-pretrained model that is trained “from scratch” are discussed. Also, the results for the two training variants mentioned in Section 3.5 by using channel dropout or only RGB information during training are shown. For channel dropout, the validation is done without channel dropout—as for the other methods—in order to preserve comparability. Some performance plots are shown in Figure 3.11. During the training process, all training loss components converge to similar values, whereby the pretrained models converge, as expected, substantially faster. The validation loss shows similar behavior. In addition, a slight overfitting can be observed for all trainings, as the loss is increasing after reaching a minimum at about epochs 10 – 15 for the pretrained and about epochs 50 – 80 for the non-pretrained models. The overfitting cannot be observed in terms of validation MDO. For both validation loss and validation MDO the pretrained models reach, besides the faster convergence, slightly better values compared to the non-pretrained models. Overall, the convergence for the channel dropout training is tendentially slower than for the RGB-only and even slower compared to the full model. However, there is

no significant benefit visible of using channel dropout during training. Apparently, the model already uses cross-channel information sufficiently well. “Distracting” the model from single-channel traits by randomly canceling out single-channel information seems to be not required.

All in all, a stable and well-performing preprocessing method is determined and has assured that choosing a ViT backbone and pretraining the model on environmental data is beneficial for the model performance. The model has the capability to use the multispectral information across channels and is able to learn the low-level features of the plant images. However, the model is only evaluated on the validation dataset so far which is, being a random subset of the training data, quite similar to the training data. In a next step, the SugarViT models are evaluated on unseen test data that is completely unknown by the model, in order to see the generalization capabilities of the approach.

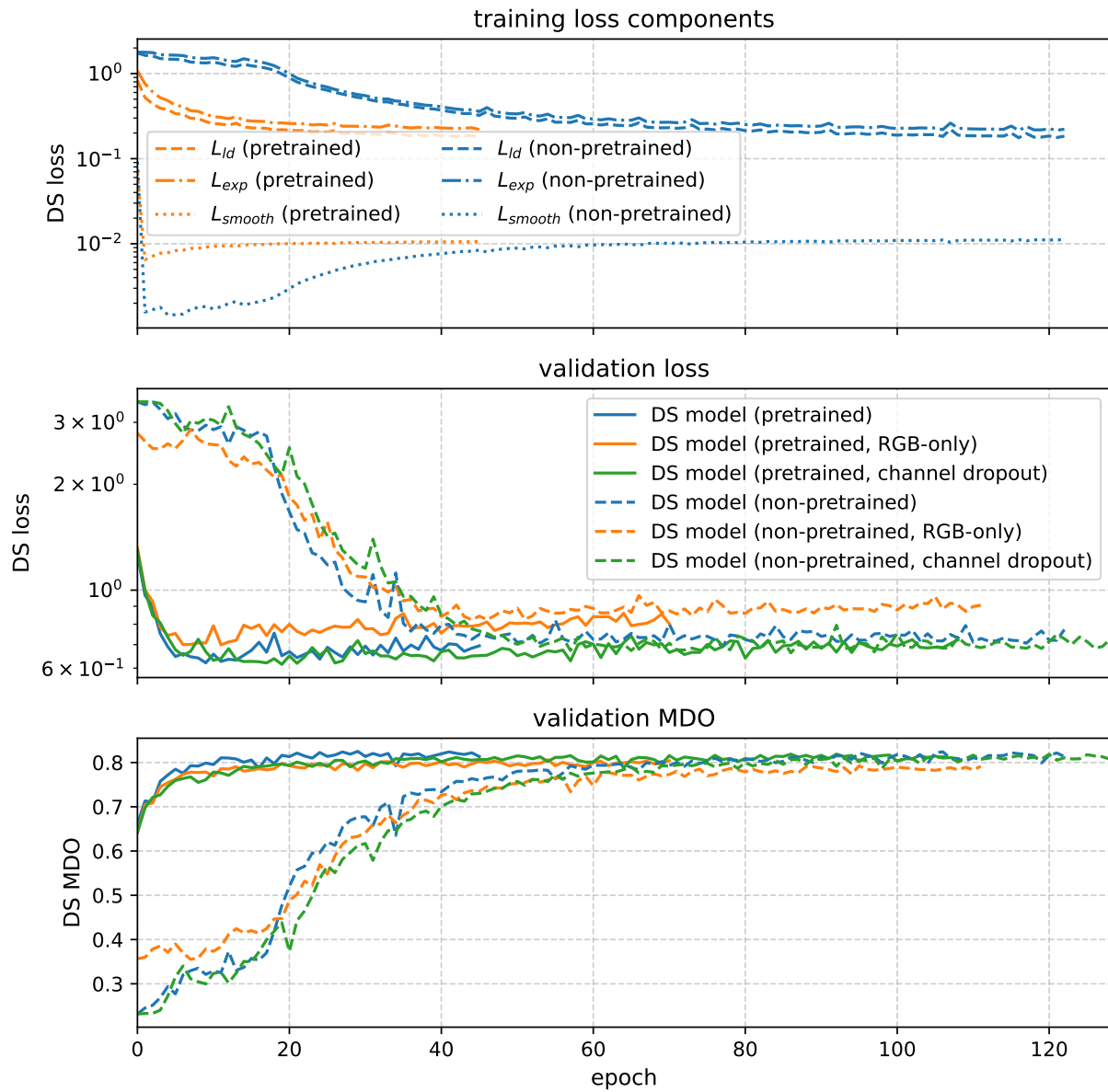


Figure 3.11: Comparison of the pretrained and non-pretrained SugarViT model. The top plot shows training loss by epoch separated into the 3 loss terms (cf. Equation (3.8)). Blue lines show the loss components by epoch for the non-pretrained model, orange lines show the ones of the pretrained SugarViT model. The two plots below show metrics, namely validation loss (center) and validation MDO (bottom), against the training epoch. Solid lines show the pretrained, dashed lines the non-pretrained model. The colors denote the three different model variants.

3.5.4 Evaluation on the Test Dataset

Conclusively, the test dataset D07b (cf. Table 1.3) is used to evaluate SugarViT on unseen data. Although stated that the MAE is not an appropriate metric for the DLDDL approach in SugarViT, it can give some insights on the prediction quality in the field usage, since the expectation value of the predicted label distribution is used as the overall model prediction. When analyzing large numbers of plants, the actual predicted label distribution is not as interesting as the actual expected DS value. The purpose of the distribution itself is more of interpretational nature in order to find weaknesses of the model predicting the plant traits of choice. Thus, recap that the MAE of N DS predictions is given by

$$\begin{aligned} \text{MAE}(\text{DS}) &= \frac{1}{N} \sum_{i=1}^N \left| \text{DS}_{\text{true}}(\mathbf{x}_i) - \text{DS}_{\text{pred}}(\mathbf{x}_i) \right| \\ &= \frac{1}{N} \sum_{i=1}^N \left| \mathbb{E} [\mathbb{P}(y \mid \mathbf{x}_i)] - \mathbb{E} [\hat{\mathbb{P}}(y \mid \mathbf{x}_i)] \right|. \end{aligned} \quad (3.11)$$

Three model adaptations, i.e., with using all information, with using channel dropout during training, and with only using RGB channels, are evaluated; each of them with and without using pretraining. Certainly, recognitions can still be incorrect. However, one can apply some techniques to reduce the errors. For instance, the input can be augmented and the average of all augmented inputs is used as the final prediction for the augmented image. One could use any augmentation that is also used during training. However, “simple” augmentations like mirroring and rotation in order not to reduce the performance, i.e., inference time or usage of computational resources too much should be enough. Thus, one single image is augmented into 8 instances in total. Since the model outputs are PMFs, one can just add them and renormalize them by dividing by 8. Please note that using this augmented evaluation is always a tradeoff between prediction robustness and inference time or computational effort. Thus, it is only recommended to use this technique, if enough resources are available in order to guarantee a performant workflow.

As stated in Section 3.5.3, overfitting is observed in the validation loss, while the MDO is still increasing. Therefore, the training variants are evaluated by their best models with regard to both metrics separately. It can be observed that the models with the best validation MDO, respectively, perform slightly better in test data evaluation. Apparently, MDO and the KLD-based loss have different objectives. Thus, the “better” model is in fact an ambiguous term. Here, we define the model with the better MAE as the better model. According to that, the MDO seems to be more sensitive to mean shifts of the distributions, while the KLD loss tends to be more sensitive on variance shifts. Those

findings indeed open a discussion of using other training loss metrics like Wasserstein distance but, however, this is beyond the scope of this chapter. Table 3.3 shows the MAE and MDO values distributed in each true DS label for the models with the best validation MDO. For comparison, Table 3.4 shows the same values for the models with the best validation loss. In all evaluations, the augmentations mentioned above are used.

The most remarkable observation is that all models show performance shortcomings for plants with intermediate DS. A reason might be the limited amount of training data in comparison to DSs that occur far more frequent. A weighted sampling as mentioned above tries to cope with this imbalance. However, the augmented data surely is not “new” data in the actual sense. Unfortunately, those intermediate DS scores are also the most ambiguous ones due to possible bias by margin of (human) interpretation and evaluation. Moreover, the importance of the two non-RGB channels is visible since the RGB-only models have a weaker performance in nearly all DS categories. This agrees with the expectation since the symptoms of the CLS disease are mostly visible in the non-optical spectrum, especially in the near-infrared channel. Thus, the NIR and REDGE channel (cf. Table 1.2) carry valuable information and the experiments show that it is indeed worth using multispectral imaging in favor of ordinary RGB imaging. Another remarkable result is that, apparently, the model channel-dropout does not show significant improvement in terms of generalization to unseen data. It outperforms the fully trained model in only a few DSs. In difference to the evaluation on validation data, the non-pretrained models outperform the pretrained ones in most cases for the test dataset. This is quite notable and opens the discussion if the pretraining possibly leads to a slight overfitting to the data and gives another hint about the importance of generalization, especially in agricultural-related use cases. Nevertheless, the idea of pretraining is still important, since training times can be reduced by adaption of one trained backbone to multiple labels of interest. Additionally, in use cases with low data coverage, finetuning existing, pretrained models might be the only way to get performant prediction models.

For the usage of SugarViT in the field, there are still some points to mention regarding further prediction improvements that are discussed in the next section.

True DS	Non-pretrained			Pretrained		
	Full	Ch. Dropout	RGB	Full	Ch. Dropout	RGB
-1	0.08	0.12	0.10	0.11	0.11	0.09
0	0.09	0.06	0.11	0.04	0.08	0.05
1	0.23	0.23	0.27	0.23	0.23	0.22
2	0.79	0.65	0.77	0.74	0.76	0.70
3	0.86	0.84	1.06	0.94	0.93	1.16
4	1.28	1.23	1.06	1.34	1.05	1.08
5	0.86	0.86	0.91	1.00	0.70	0.75
6	0.97	0.80	0.95	0.93	0.81	0.72
7	0.88	0.93	1.05	0.93	0.95	0.99
8	0.99	1.02	1.22	1.00	1.12	1.35
9	1.10	1.17	1.22	1.17	1.19	1.55
10	0.80	0.91	1.17	1.14	1.25	1.20
Total	0.46	0.43	0.50	0.46	0.45	0.46
Total (corr.)	0.74	0.73	0.82	0.80	0.76	0.82

(a) Mean absolute error (MAE).

True DS	Non-pretrained			Pretrained		
	Full	Ch. Dropout	RGB	Full	Ch. Dropout	RGB
-1	95.53	93.84	94.57	94.02	94.05	94.95
0	94.03	94.51	93.22	95.55	94.75	95.14
1	87.36	87.14	84.86	86.26	86.49	85.70
2	57.53	62.52	57.19	59.07	58.08	61.08
3	60.56	58.49	52.26	56.48	54.69	48.18
4	44.10	45.09	52.28	41.59	50.73	50.16
5	59.18	57.77	56.03	53.79	64.96	60.23
6	53.53	60.84	52.67	54.63	59.27	61.86
7	57.34	53.78	52.76	56.93	53.47	52.12
8	53.61	52.01	45.01	53.69	48.33	40.48
9	55.28	53.11	53.41	55.74	54.89	42.68
10	71.81	66.66	59.36	58.24	57.35	61.52
Total	76.89	77.53	74.64	76.28	76.50	75.83
Total (corr.)	65.74	65.48	62.80	63.83	64.75	62.84

(b) Mean distribution overlap (MDO) in %.

Table 3.3: Evaluation metrics for the model with best validation MDO on the test dataset separated by true DS value. Below the single DSs, the metrics for the full dataset are listed, both with and without correcting for the label abundance.

True DS	Non-pretrained			Pretrained		
	Full	Ch. Dropout	RGB	Full	Ch. Dropout	RGB
-1	0.11	0.11	0.15	0.15	0.13	0.15
0	0.03	0.05	0.06	0.02	0.03	0.06
1	0.43	0.35	0.33	0.34	0.22	0.33
2	0.60	0.77	0.77	0.82	0.86	0.77
3	0.83	1.01	1.29	1.09	1.11	1.29
4	1.10	1.38	1.15	1.33	1.28	1.15
5	0.78	0.76	0.79	0.83	0.76	0.79
6	0.84	0.73	0.86	0.84	0.68	0.86
7	0.90	0.84	1.01	0.85	0.86	1.01
8	1.15	1.13	1.28	1.13	1.18	1.28
9	1.24	1.15	1.48	1.25	1.32	1.48
10	1.17	1.02	1.35	1.32	1.02	1.35
Total	0.54	0.51	0.57	0.53	0.46	0.54
Total (corr.)	0.76	0.77	0.88	0.83	0.78	0.87

(a) Mean absolute error (MAE).

True DS	Non-pretrained			Pretrained		
	Full	Ch. Dropout	RGB	Full	Ch. Dropout	RGB
-1	93.41	93.47	92.63	92.39	93.06	92.75
0	95.95	95.35	93.24	95.74	95.52	95.38
1	76.67	80.97	79.28	83.49	87.05	82.01
2	66.30	53.17	56.24	56.22	54.29	58.85
3	58.58	55.21	47.08	49.23	47.41	44.30
4	49.20	40.97	46.88	41.70	42.22	49.99
5	59.67	60.63	58.11	56.14	58.42	56.27
6	57.48	62.04	58.90	56.05	63.71	54.76
7	54.66	54.78	47.97	56.14	53.15	48.59
8	47.21	45.52	43.51	47.47	44.69	42.17
9	53.11	56.11	42.79	51.24	47.89	45.32
10	58.76	62.53	49.01	52.53	63.85	55.34
Total	72.19	73.74	70.90	73.96	75.89	72.85
Total (corr.)	64.25	63.90	59.64	61.53	62.61	60.48

(b) Mean distribution overlap (MDO) in %.

Table 3.4: Evaluation metrics for the model with best validation loss on the test dataset separated by true DS value. Below the single DSs, the metrics for the full dataset are listed, both with and without correcting for the label abundance.

3.5.5 Application in the Field

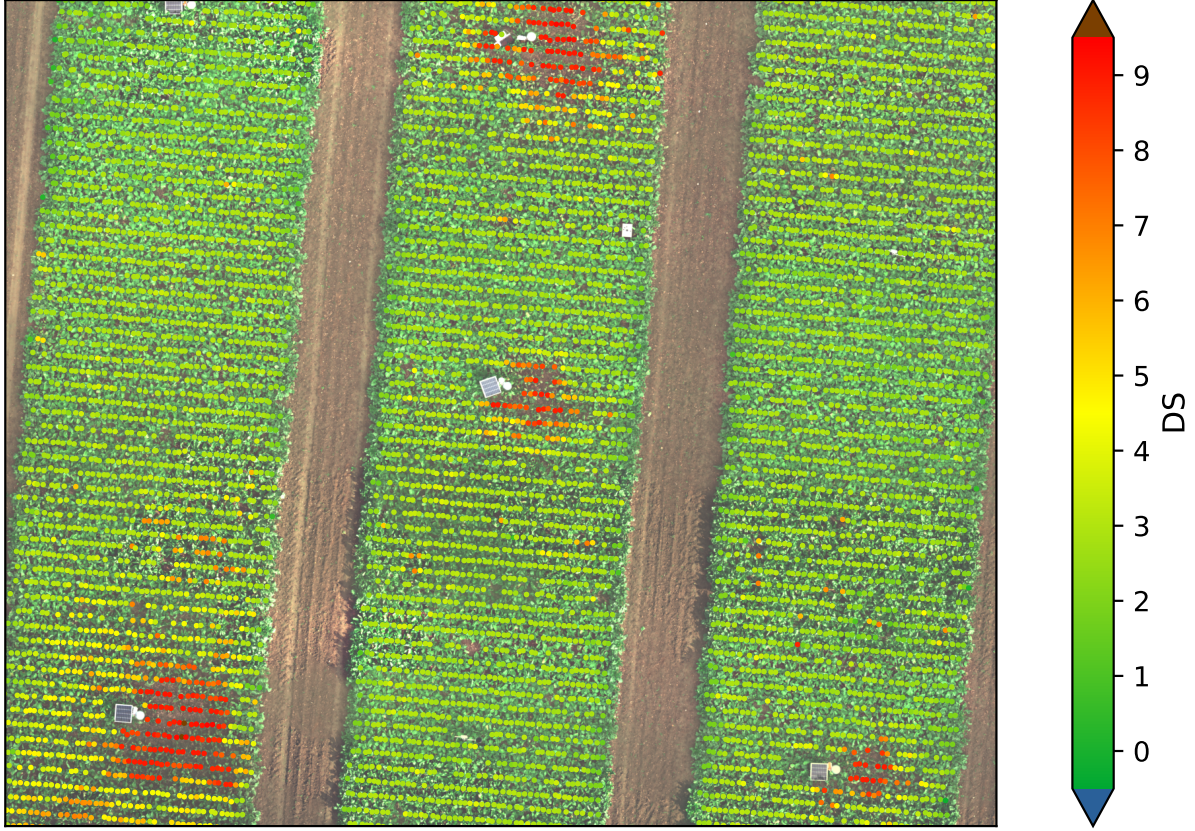


Figure 3.12: Exemplary application of SugarViT for disease severity prediction on unseen UAV data. Each prediction is completely independent of its surrounding predictions. The model shows a highly consistent prediction behavior.

Frameworks like the one presented in Chapter 2 help to extract the plant positions and single images for large-scale UAV image data. Thereafter, the trained SugarViT model can be applied for new field experiments, enabling fast large-scale DS annotations.

Figure 3.12 shows an exemplary application of SugarViT to a test dataset image using the described augmented evaluation. However, the model does not consider temporal and spatial dependency between the plant images so far. The error rate could further be reduced by correcting single “obvious” outliers that do not fit into the temporal and spatial vicinity of the other plants. Additionally, SugarViT has the advantage to actually output a label distribution. Since the (fixed) training label std σ_{train} is known for the trained model, one can compare the std of the output (assuming a normal distribution) σ_{pred} with it in order to see, how “confident” the model is in its prediction. Thus, for each output, a “confidence”

$$\tilde{c} := \frac{\sigma_{\text{pred}}}{\sigma_{\text{train}}}, \quad (3.12)$$

that is 1 for an exact conformity of stds can be defined. For values $\tilde{c} > 1$, the model is more and more unconfident, whereas for values $0 < \tilde{c} < 1$, the model is over-confident in its decision. However, please note that this is no proper confidence in a statistical sense, since the expectation value could still be completely wrong. Since the expectation value in an unlabeled dataset is unknown, this purely std-based value is a rather imprecise yet helpful measure of the model’s confidence. Nevertheless, this value can serve as a “warning message” in a deployed model that indicates the necessity to cross-check the model output with the estimation of a human expert. The expert could then correct the model output and complicated examples can be identified and used for retraining the model. Thus, the confidence measure and, in the end, LDL enables the possibility for a continuous learning process which is commonly referred to as *Active Learning* with human feedback.

In order to provide integrity and usability of the model in the field, it is important to have interfaces to widely used software and data formats. For instance, the libraries *GeoPandas* and *rasterio* can be used to load orthoimages and plant positions as *geopackage* or similar files in order to evaluate models on them. Afterward, the prediction can be exported again to *geopackage* format or, for instance, as *Pandas DataFrame* objects. Thus, an interface to widely used geographic information system (GIS) software that is frequently used for georeferenced image data is provided.

3.5.6 Attention Maps

Due to the fact that the backbone of SugarViT is based on the attention mechanism (cf. Section 1.2), one can analyze and—ideally—interpret which image features are more or less important to the model’s decisions. As already described in Section 1.2, one helpful visualization method for that are so-called *attention maps*. Roughly spoken, attention maps can visualize, “where the model looks at”. The ViT backbone in SugarViT consists of 8 attention layers, with 4 attention heads each that can in principle be trained to focus on completely different features. In order to accumulate the attention maps of each single layer to one overall map, the technique of *attention rollout* [1] (cf. Section 1.2.5.3) is used. Figure 3.13 shows the result for one randomly chosen image from the validation dataset per DS class. A main observation is that SugarViT indeed focuses on the plant itself and not, e.g., on the amount of visible soil around it. This is particularly visible for the DS 0 example. Additionally, one observes that the model focuses on multiple image regions that seem to be complementary for the decision-making process, which is exactly the power of the attention mechanism compared to CNNs. CNNs learn filters that are applied on the whole image. Thus, relations between pixel values can only be covered locally. The attention mechanism allows connecting those local features with other

distant image regions and, thus, is considered to have more power of “understanding” the image as a whole.

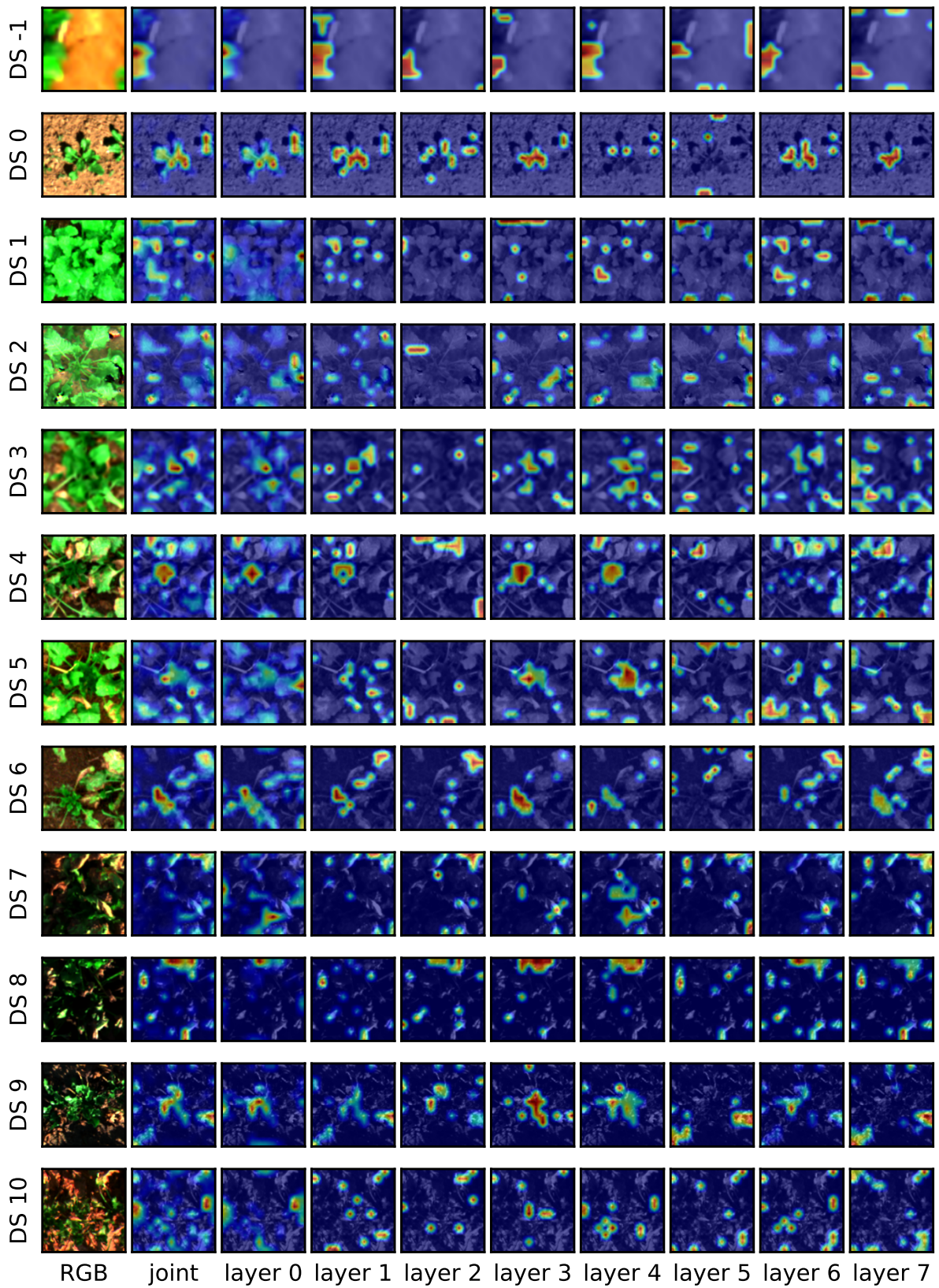


Figure 3.13: Attention maps for an example image per DS class. The first column shows the original input image in its RGB representation. The second column is the joint attention map after performing attention rollout [1]. The following columns are the attention maps of each of the 8 attention layers in SugarViT.

3.6 Discussion

This chapter shows that simple normalizing methods like standardization can already outperform more sophisticated and expensive normalization methods like histogram equalization (cf. Section 3.5.1). For the DS estimation use case—where spectral differences carry much information—a total standardization leads to better results than a channel-wise standardization. However, it has to be stated that the image data has been calibrated by a reflectance panel during acquisition, so the spectral information is directly comparable. Thus, if the data cannot be calibrated for any reason, the channel-wise standardization may be a good choice, since it also leads to acceptable prediction qualities.

The comparison experiments have further revealed that—if enough data is given—using a ViT backbone is a good choice. However, for lower data availabilities, also convolution-based backbone networks can reach comparable performances. Furthermore, it is observed that beyond-RGB imagery is beneficial for the prediction of DS in sugar beet plants. This supports the findings of [89] and shows that image information in non-optical bands improves the prediction quality. The ViT backbone is able to use the additional information since the use of channel dropout during training does not show significant improvements in the prediction quality.

Lastly, the pretraining on environmental metadata turns out to be beneficial for the final prediction quality as well as the training speed (cf. Section 3.5.2). Accumulating environment-related features like GDD and NPG contains information regarding the plant growth stage and the disease stage, respectively, while being robust against seasonally varying weather conditions. Thereby, different harvest seasons with very contrasting weather conditions become comparable. Additionally, the pretraining, environmental annotation, and the subsequent finetuning on the annotation of interest can be an approach for further generalization even on smaller datasets. The pretrained ViT backbone can be seen as a fixed, plant-image-aware feature extractor that has learned plant specific traits. On top of that, a smaller model can be trained for different annotation purposes, which accelerates and improves the training procedure substantially. Nowadays, this concept is used in so-called foundation models (FMs) where a large, versatile model is trained to recognize general patterns and is used in a multi-task fashion. For specific tasks where the model size of the FM would exceed computational resources for local training, smaller models can be trained on the basis of the large model. Pretraining also enables the usage of large-scale image data. Even if, for instance, only few data is labeled with expensive human-expert annotations, the unlabeled majority of the data can still be used in a pretraining stage unfolding the full potential of the collected data. If data labeling originated from multiple human experts that may have ambiguous estimations

or assessment guidelines, the usage of LDL is able to incorporate detailed uncertainty information in training process and, finally, in the model.

The use of attention mechanism instead of convolutional networks turns out to make sense in this use case because of the long-distance relations of leaf spots. The interpretability of resulting attention maps on single instances is often questionable. However, they can reveal if the model focuses on the right regions and features itself rather than on spurious correlations.

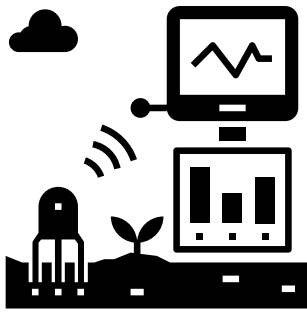
3.7 Interim Conclusion

Generalization to data from unseen fields with unseen weather conditions and climate is certainly one of the most challenging questions in data-driven machine learning approaches in agriculture. The findings in Section 3.5.4 emphasize that. The available data in this work already comprises 4 growing seasons but only from some locations in Central Germany. Whether the model performs comparably well in other regions is at least questionable. Nevertheless, a model that is at least locally accurate already has a high value for increasing the efficiency of DS assessment. Extrapolation to other environmental conditions is challenging, but interpolation on the same field has the potential to save valuable expert working time in the field, where the model can complement a few spot-wise expert annotations on the whole field. As seen in Section 3.5.4, the high data imbalance and label ambiguity still remain challenging, even with weighed sampling and LDL, respectively. The disease assessment in individual objects as plants is hard to standardize and schematize by exemplary images as, for instance, in the case of CLS. Therefore, it is important to have a model that incorporates label uncertainties and is transparent in its prediction uncertainties, like in SugarViT.

With this large-scale DS assessment available, further challenges regarding disease assessment can be tackled. The retrieval of DS complemented by further environmental sensors enables, for instance, detailed investigations on disease spread and its modeling. Consequently, the approach could also find application in terms of disease control by, for instance, punctual application of pesticides, lowering costs and environmental impact. Perspectively, the concept behind SugarViT could also be applied in a wide variety of other use cases in the field of UAV-supported phenotyping. [7, 25, F, 160]

If SugarViT is seen as a general framework for plant-related image-based regression tasks, the model can be adapted to a wide variety of use cases in precision agriculture. Indeed, there is potential to extend SugarViT to other crops and diseases that would make it sufficiently versatile to be called an FM. This could be some follow-up work if enough data is available. Additionally, the model could be extended to a multi-task

learning approach where multiple annotations of interest are predicted at once. Thereby, SugarViT could profit from its interrelation facilities.



4

Disease Spread Modeling

The concept of this chapter is inspired by the publication [B]. However, many parts of it have been significantly reworked and extended for this dissertation. In the previous publication, my contribution included the conceptualization of the disease spread modeling workflow, the design and implementation of the software, conducting the experiments, and writing the manuscript. The publication was a collaboration with colleagues from the Institute of Sugar Beet Research (IfZ) in Göttingen, where the data was acquired and prepared.

4.1 Introduction

Over many years of agricultural research, extensive knowledge of phytopathology and plant epidemiology has been accumulated. [86] For efficient pesticide application and, thus, sustainable handling of environmental resources, precise knowledge about crop diseases and their dynamics is essential.

In the past, many small-scale, empirical studies in controlled environments like [12–14] were conducted. They serve to improve understanding of the pathogen and to establish a relationship with easily measurable parameters such as temperature, humidity, etc. However, controlled environments are a substantial simplification of reality that comes with biases and errors which can now, at least in part, be addressed with parameters obtained by remote sensing (RS). This technology furthermore allows researchers

to consider spatial variables and incorporate the heterogeneity of environments. [F, 7, 101, 118] Accompanied by the progress in computational resources, data analytics, and the success of artificial intelligence (AI) in many disciplines, the field commonly known as *precision agriculture* was born. [100, 136] Nowadays, mostly classical machine learning (ML) methods like support vector machines (SVMs) and Random Forests, but also some more advanced methods like convolutional neural networks (CNNs) are used in a wide range of agricultural applications focusing on RS and geoscience. [6, 78]

In phytopathology, however, one of the major challenges is the complexity of interactions between plant organism, pathogen, pesticide, soil, climate, etc. Many processes are difficult or impossible to observe directly or show unpredictable behavior, especially when transferring insights from controlled/in-vitro environments to real-world/in-vivo applications. This is why phytopathological models need to incorporate lots of diverse observables and parameters to yield precise outcomes.

Although the approach is rather generic, it is discussed here in the context of the disease incidence and spread modeling of the fungal disease *Cercospora* leaf spot (CLS) in sugar beet field experiments. CLS—caused by the fungus *Cercospora beticola* Sacc.—is the most important and impacting foliar disease on sugar beet [119, 148]. This is why various studies have been conducted about spread modeling and prediction of CLS on sugar beet [34, 145, 155]. Additionally, so-called negative prognosis models have already been developed and are used for the prediction of CLS infections based on environmental and easily accessible field data [115, 124]. In those infection models, many results of the empirical studies and systematic correlations are successfully used. However, the mentioned studies lack an analysis of the heterogeneity of the microclimate and none of them consider information on the landform configuration and the state of development of the plants.

This chapter aims to leverage contemporary ML methods to enhance pure simulation models. It proposes a workflow and a modeling approach that unifies an agent-based simulation driven by empirical phytopathological knowledge with a data-driven ML model. In the end, the model should simulate a field and the disease spread dynamics of generic plant-pathogen-combinations. Furthermore, the workflow facilitates the use of unmanned aerial vehicle (UAV) image data for the generation of large-scale epidemiological datasets. Together with other data extraction and learning workflows from UAV imagery as in Chapters 2 and 3, one can use large-scale data for precise prediction of disease spread dynamics and adaptation to local conditions, while having flexible models that can be generalized to various plants and diseases. An important first step towards this goal has already been made in [H], where a hybrid modeling approach combining mechanistic and data-driven methods for CLS epidemiology is elaborated. Now, this work is extended by a more sophisticated ML model architecture and a more generic modeling approach.

The following spread modeling makes use of the data introduced in Section 1.5. Precisely, the datasets D03, D06, D07, D08, and D09 are used. For more details on the data, see Table 1.3. Additionally, the results from the previous chapter about disease severity prediction using the SugarViT model will serve as input for the spread modeling.

4.2 Plant-specific Parameters from Empirical Knowledge

In addition to image-based data acquisition, which is relatively novel compared to the long history of agricultural research, there is a large body of empirical knowledge that has been available for decades. This work explicitly wants to make use of these data sources to guide and support the machine learning techniques. Today, many phytological and phytopathological processes are well-understood, even if their particular observation usually is challenging.

Firstly, the concept of *thermal time* [84] is used. Non-plant-specific measures like growing degree day (GDD) [28] or the reference evapotranspiration (ET_o) [3] can be calculated by using the environmental data (cf. Table 1.1). For GDD, the calculation from Section 3.2.1 and Equation (3.1) is reused. As an extension to that, the ET_o is a more sophisticated measure including more weather parameters to estimate the net evapotranspiration of a reference surface. It describes the total water consumption of a plant, considering evaporation and transpiration. Since it uses a reference surface, it is a non-crop-specific parameter for plant growth and can be estimated by the so-called Penman-Monteith equation. [3]

Another usable high-level parameter is the plant development stage in BBCH scale [94, 103, 164]. BBCH is an acronym of Biologische Bundesanstalt für Land- und Forstwirtschaft, Bundessortenamt und Chemische Industrie and is a standardized scale to describe the phenological development of plants. It is structured in principal growth stages and secondary growth stages. For sugar beet, the principal growth stages range from 0 (germination) to 9 (senescence). Each principal growth stage is further divided into secondary growth stages ranging from 0 to 9, resulting in a two-digit code. Since the plant development is highly dependent on environmental conditions, one can estimate the development stage by using the GDD [56, 103]. For numerical reasons, the discrete classes are transferred into a continuous scale by using a cubic spline interpolation. Figure 4.2 shows the relation between GDD and the BBCH development stage for sugar beet.

These parameters are used cumulatively, i.e., they are summed up from the day of sowing until the actual day. That way, they can be treated like a “phytological time”, since they account for environmental conditions throughout the harvest season. Thus, different harvest seasons of years with different environmental conditions become more

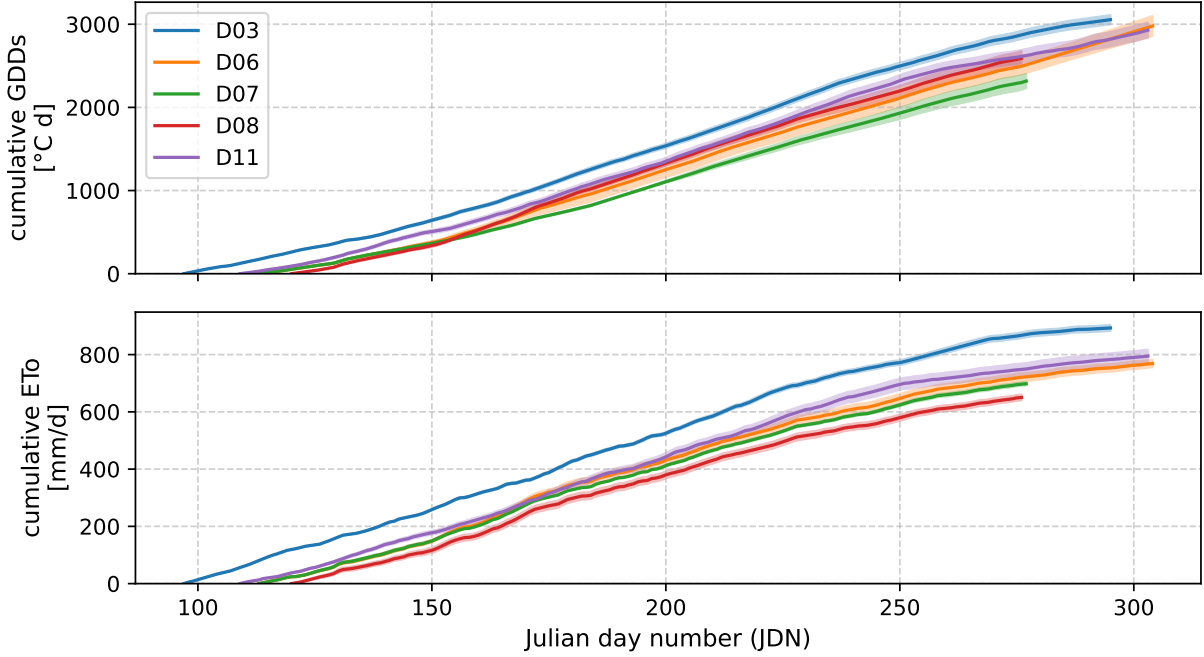


Figure 4.1: Comparison between Julian day number (JDN) and the growing degree day (GDD) (top) and reference evapotranspiration (ETo) (bottom) used as “phenological time”. Different lines represent the daily mean of all datasets used in this work. The transparent bands represent one standard deviation (std). The starting point of the accumulation of GDD and ETo is different among the datasets, which is why the curves start at different JDN. However, the slopes of the curves are quite similar, making “phenological time” represented by GDD and ETo a well-suited substitute for physical time.

comparable. This is why physical time is replaced with those measures for the analyses in this work. Figure 4.1 shows the relation between the Julian day number (JDN) and the parameters GDD and ETo. One can clearly see that the starting day of accumulation is different among the datasets and years. This is due to different weather conditions and sowing dates. However, the slope of the cumulative GDD and ETo is quite similar among the datasets, which means that the environmental conditions are comparable when using those measures instead of physical time.

Secondly, there are plant-specific parameters that mostly account for the potential phenological development, for instance, growth stage, of individual plants. From Section 2.4.2.3, one can derive the soil cover ratio of the field and transfer it to the single plant images. It serves as a proxy for the plant size and, thus, its growth stage, especially before canopy closure, which is the time when 90% of beet plants in adjacent rows begin to touch or overlap [157]. From this value, one can then derive the plant radius, if the sowing distance is given. Here, it is $d = 24$ cm for all experiments, so that the plant radius r_p can be calculated by

$$r_p(t) := \frac{d \cdot c(t)}{95\%}, \quad (4.1)$$

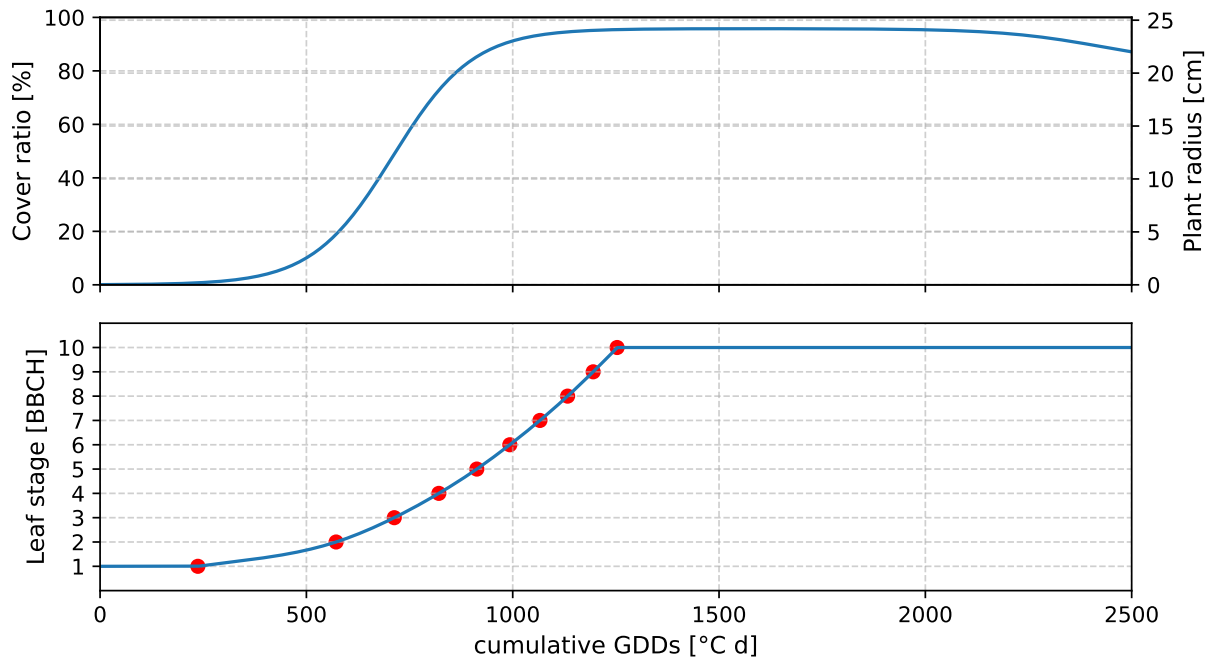


Figure 4.2: Relations of cover ratio, plant radius (top), and leaf stage (bottom) versus growing degree day (GDD). For the cover ratio curve (top) and the conversion to plant radius, the growth function proposed in Section 2.4.2.3 is used. The red dots mark the transition points of the leaf stage classes given in [103] (bottom). Both values can be simulated stepwise for each day, since the cumulative GDD or ETo can be calculated by each day's environmental parameters.

where $c(t)$ is the cover ratio function (cf. Equation (2.5)). The division by 95 % is due to the empirically found maximum cover ratio in the regression experiment. Another parameter aiming to quantify the plant development in the so-called leaf stage of sugar beet [56]. It is represented by classes between V1 and V10. Definition details of the class definitions can be found at [103]. In this work, the discrete classes are converted into a continuous scale by spline interpolation of its relation to GDD given in [103]. Figure 4.2 shows the relation of cover ratio, plant radius, and leaf stage with GDD.

Thirdly, disease-specific parameters like infection probabilities, incubation and fructification periods, as well as daily spore production (DSP) for CLS investigated mainly in controlled environments are available from various sources [12, 13, 124, 154, 155]. Two essential time periods in the disease development of CLS are the incubation and fructification time. The incubation time is defined as the time period from the infection to the time, when first symptoms are visible. The fructification time is defined as the time from the infection to the time, where the conidia (or spores) of the fungus begin to spread [12]. For the use with variable environmental conditions, the absolute incubation and fructification times are translated into *cycles*. The principle is as follows: let $T(d, h)$ be the temperature of a day d and each hour h . For each of these hours, the inverses of the incubation and fructification time $t_{\text{inc/fruct}}(T(d, h))$ are calculated and accumulated.

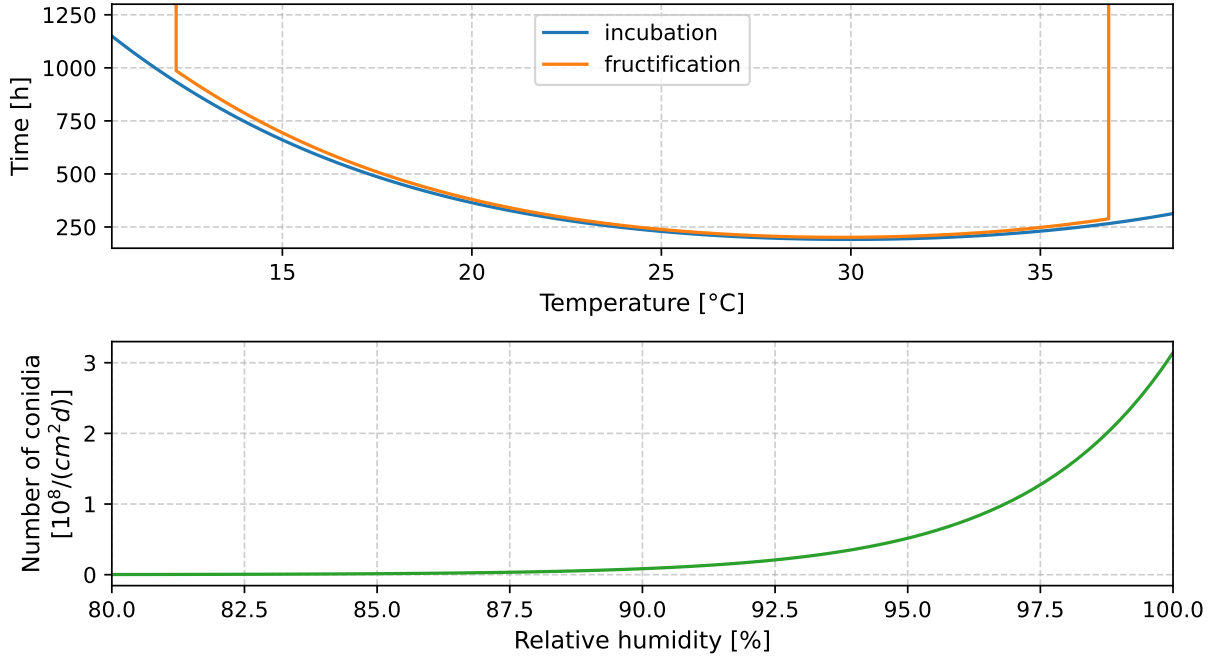


Figure 4.3: Functional relations between temperature, relative humidity, and disease development. The above plot shows incubation and fructification time versus temperature from [12]. Below 12 °C and above 37 °C, fructification was not observed. Thus, the fructification time is infinite in these domains. The bottom plot shows the relation between relative humidity and the spore production from [13]. The spore production increases exponentially with relative humidity. For determining the daily spore production (DSP), the hourly values are averaged over the day.

One incubation/fructification cycle is completed when the sum adds up to 1. Hence, for the incubation/fructification cycle $\zeta_{\text{inc/fruct}}(d)$ at day d , it is

$$\zeta_{\text{inc/fruct}}(d) := \sum_{i=1}^d \sum_{h=1}^{24} \frac{1}{t_{\text{inc/fruct}}(T(d, h))}. \quad (4.2)$$

Figure 4.3 shows the used functional relations between the temperature and the fructification and incubation time as well as the relation between relative humidity and the DSP. Moreover, [154] introduced a daily infection value (DIV) that counts the number of hours with a relative humidity $\geq 90\%$ and uses the average temperature during those hours to calculate a risk category for the considered day. The DIV will be used as an input for the model as well. Table 4.1 summarizes all the used plant parameters along with references for their calculation.

Parameter (Symbol)	Unit	Reference(s)
Growing degree day (GDD)	°C d	[56, 103]
Reference evapotranspiration (ET _o)	mm/d	[3]
Sugar beet plant development stage	ordinal, BBCH	[56, 94, 103, 164]
Estimated cover ratio (c)	%	derived from Chapter 2
Estimated plant radius (r_p)	cm	derived from Chapter 2
Disease severity (DS)	ordinal	[E]
Incubation period / cycle (ζ_{inc})	h / -	[12], Equation (4.2)
Fructification period / cycle (ζ_{fruct})	h / -	[12], Equation (4.2)
Daily infection value (DIV)	ordinal	[154]
Daily spore production (DSP)	1/(cm ² d)	[13]

Table 4.1: Plant parameters structured by non-plant-specific (top), plant-specific (middle) and disease-specific (bottom) variables. The right column refers to the source(s) used to estimate or calculate the respective parameter.

4.3 Estimation of Infection Date

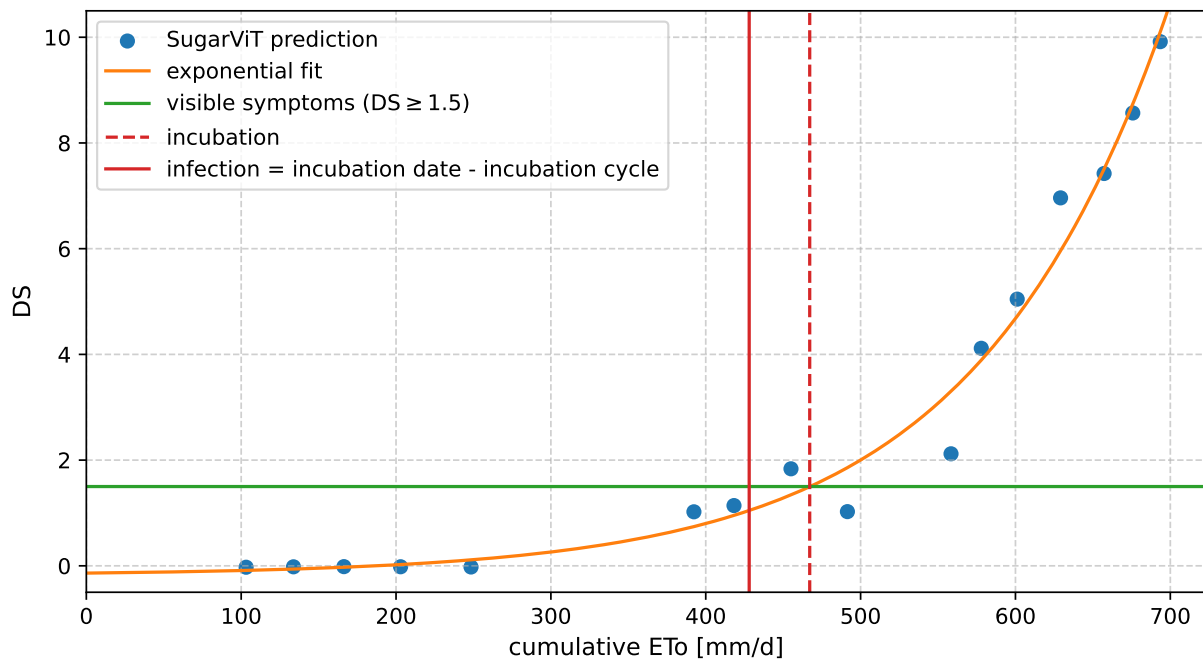


Figure 4.4: Exemplary visualization for the retrieval of the infection date. The punctual severity predictions of the SugarViT model (cf. Chapter 3) (blue dots) are fitted by an exponential function (orange line). The intersection with a disease severity of 1.5 (green line) is identified as the incubation time (red dashed line). Starting from this point, one retrieves the infection point (red solid line) by backtracking the incubation cycle.

Obviously, for training an infection model, one needs to have the infection day of individual plants. However, the date of infection is not visible directly. Visible symptoms of the CLS disease only emerge after the incubation period has passed. Thus, the visual severity annotation from Chapter 3 can help identify the incubation date. Because the

incubation cycle is tracked, one can propagate back to retrieve the infection date. Consequently, the retrieval strategy is as follows: first, take the disease severity (DS) prediction from SugarViT and perform an exponential fit with the function

$$DS(t) := A \exp \frac{t}{\tau} + b, \quad (4.3)$$

where A , τ , and b are fit parameters. First visible symptoms of CLS can be seen between DS 1 and 2. Therefore, the *incubation date* t_{inc} is determined by

$$t_{\text{inc}} = \tau \log \frac{1.5 - b}{A}. \quad (4.4)$$

Counting the incubation cycle from the day of sowing—i.e. pretending that the plant is infected at its sowing—one evaluates the cycle values at the incubation date and subtracts one cycle. Hence, at the infection date t_{inf} , it is

$$\zeta_{\text{inc}}(t_{\text{inf}}) = \zeta_{\text{inc}}(t_{\text{inc}}) - 1. \quad (4.5)$$

Given that, the date where ζ_{inc} is closest to the calculated value is defined as the date of infection. Note that the JDN is taken here again. Figure 4.4 summarizes the infection date retrieval strategy visually.

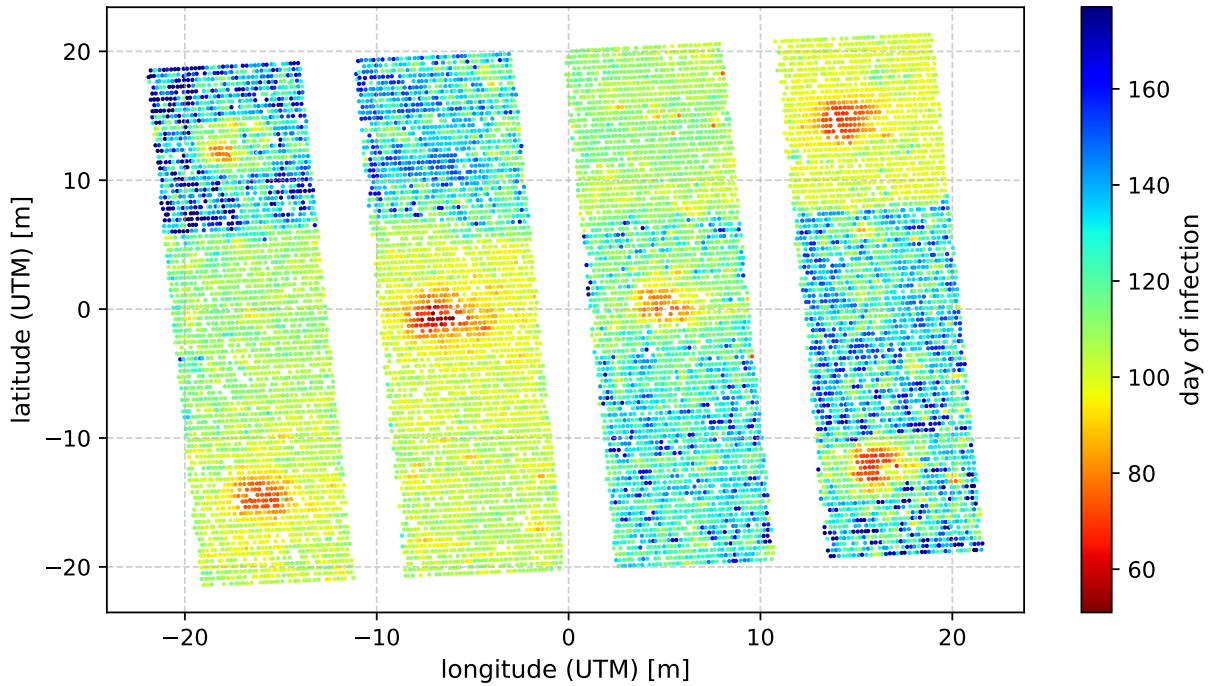


Figure 4.5: Infection map for the D06 dataset. The plant positions are colored by their day of infection. The sowing day is considered $d = 0$. The coordinates are given in universal transverse Mercator (UTM) coordinate system centered at the experimental field.

Performing this calculation for each plant in the field experiments results in a first estimation of the daily disease spread dynamics. It can be nicely summarized by an infection map showing the plant positions colored with the corresponding infection date. Figure 4.5 shows such an infection map for the D06 dataset. One can even see the different treatments of the—in this case—12 subplots.

4.4 Gathering the Infection Data

The problem of the hidden infection date is solved and the temporal infection dynamic of the plants is unraveled. In the following, the plant that is to be examined for possible infection is called the *ego plant*. Now, the challenge is to identify plant-to-plant infection data from the infection maps. Simply put, one cannot know with certainty which *neighbor plant* in particular infects the ego plant from raw data. Moreover, there is also the possibility that the ego plant gets infected without any infected neighbors. A reason for this might be that the pathogen is already in the soil—from previous seasons or due to inoculation in experiment sites—and the environmental conditions are favorable for an infection, i.e. high *infection pressure*. By construction, the simulation model knows the incubation and fructification cycles of each plant agent. Thus, the information which neighbor plants are in which infection stage is inherently available. Given the infection maps one can iteratively generate datasets for an ML training by extracting the environmental data at the ego plant and the relevant plant-specific parameters for ego and corresponding neighbor plants.

Of course, the spatial structure of the field has to be mapped in the data as well. Since all locations of the plants including their elevation are available, one can calculate the relative position vectors of each neighbor plant with respect to the ego plant. This way, the model can gain a spatial understanding by using information about the spatial distribution of the neighbor plants. All in all, the following data are gathered for the infection model:

- **Hourly** environmental data at the ego plant at day d .
Temperature [°C], relative humidity [%], wind direction [°], wind speed [m/s], precipitation [mm], global radiation [W/m²], leaf wetness [%], spore concentration [ng]
- **Daily** plant-specific parameters of the ego plant at day d .
Plant development stage [interpolated BBCH as function of GDD], cover ratio [%], reference evapotranspiration (ET₀) [mm/d], growing degree day (GDD) [°C d], Plant radius [cm], Number of fungicide applications until day d [count], daily infection value (DIV) [ordinal]

- **Daily** plant-specific parameters of the neighbor plants at day d .
Plant development stage [interpolated BBCH as function of GDD], cover ratio [%], reference evapotranspiration (ET_o) [mm/d], growing degree day (GDD) [°C d], Plant radius [cm], Number of fungicide applications until day d [count], Cercospora leaf spot (CLS) incubation cycle [cycles], Cercospora leaf spot (CLS) fructification cycle [cycles], daily spore production (DSP) [$1/\text{cm}^2$]
- Relative coordinates of the neighbor plants with respect to the ego plant [m].
- Binary infection label whether the ego plant got infected on day d or not.

In the next section, the infection modeling approach is concretized systematically.

4.5 Agent-based Infection Simulation Model

From the insights above, it becomes clear that at least some processes inside single plants are well described and readily quantifiable. However, interactions of the plant with many other influencing factors like soil, environment, other plants, etc. are complex which makes the modeling of infection probabilities challenging. As an initial modeling step, the modeling approach is structured as an agent-based setup sketched in Figure 4.6. Each plant is an agent in a field which itself is part of the environment and has a fixed location and parameters that are inferred by sensor measurements. These sensors are also located in the field, have a location, and measure a certain quantity. Their measurements are steered by the weather, which is part of the environment as well. A disease is modeled as an attribute that a plant can have. It has its own parameters that are also measured or calculated

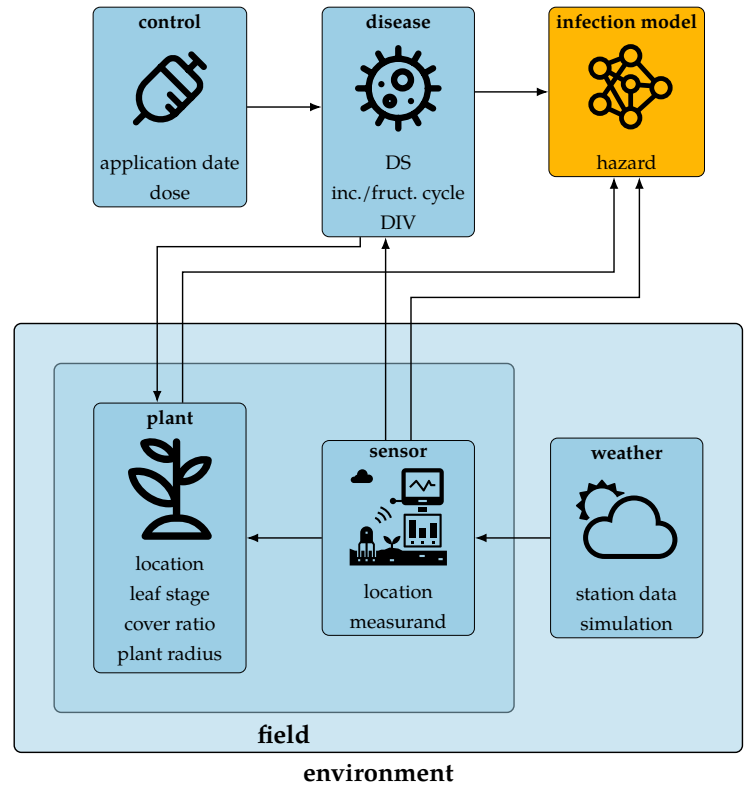


Figure 4.6: Generic interaction principle of the modeling approach. The plants are agents in a field and an environment that get sensor measurements and can have a (controlled) disease. Beside these parts of the model that can be simulated, machine learning (ML) is used to build an infection model that learns the complex interactions between plants, disease, and environment to predict the infection hazard.

by sensors. Moreover, a disease control agent can influence the disease in its severity or spread dynamics.

Up to this point, all agents have deterministic attributes that can be calculated or accumulated for known (local) environmental conditions. So theoretically, all those agents are simulatable in a certain setup. The interaction point that is not easily simulatable is the infection model itself. In the modeling paradigm, the infection model gets plant parameters and sensor measurements in order to calculate an infection probability. Once having this probability, one can again simulate a random process and, if an infection happens, instantiate the disease on the newly infected plant. Furthermore, multiple infections are not assumed to make the disease more severe. Hence, once a plant has been infected, it cannot be infected with the same disease again.

4.6 Infection Hazard Modeling

Now that the modeling approach is structured, the infection model is concretized mathematically. First of all, a set of rules and nomenclature for the infection model is defined as follows.

- R1 The ego plant has a limited spatial neighborhood choosing the K nearest neighbors by Euclidean distance.
- R2 The infection model should consider a temporal window by choosing a reversed-time window of D latest days starting from the current day d going to the past including “today”. Hence, the window is $\mathcal{D}_d := \{d, d-1, \dots, d-D+1\}$. For $d < D$, the window is truncated or padded accordingly. This modeling assumption is known in statistics as finite-memory, or *D-step Markov*, assumption [102, 116]. Particularly in plant disease modeling, it has been shown that infection events mostly depend on recent environmental conditions [53, 71, 124, 154] justifying this assumption of a limited window. Furthermore, let $\mathcal{D}_{\text{tot}} := \{1, 2, \dots, d_{\text{max}}\}$ be the complete observation period with d_{max} being the last day of the field experiment.
- R3 Once infected, the plant cannot be infected again. Thus, for the infection date $d_{\text{inf}}^{(i)}$ of the i -th ego plant, only the days $d \leq d_{\text{inf}}^{(i)}$ are considered, so that the infection day $d_{\text{inf}}^{(i)}$ is always the first day in the reversed-time window. If the plant never got infected, it is $d < d_{\text{inf}}^{(i)}$. In survival analysis [57], this is commonly known as *right-censoring* [74]. Given this rule, let further $\mathcal{D}_{\text{valid}}^{(i)} := \{d \in \mathcal{D}_{\text{tot}} \mid d \leq d_{\text{inf}}^{(i)}\} = \{d_{\text{inf}}^{(i)}, d_{\text{inf}}^{(i)} - 1, \dots, 1\}$ be the set of valid days for the i -th ego plant.
- R4 The infection model may not use future data beyond the queried day d . It should predict the infection hazards given today and $D-1$ previous days. This is referred

to as *nowcasting*. By combination with Rule R3 this implies that the model is only applied on healthy ego plants.

- R5 Neighbor plants are not filtered by their infection/health stage. The model should learn, which neighbors are relevant for an infection and which are not. Hence, it is ensured that the model can handle spontaneous infections as well as forced infections by neighbors.
- R6 The infection event follows a Bernoulli process, i.e., the plant gets infected at day d with probability p and stays healthy with probability $1 - p$.
- R7 The infection model should output a daily infection hazard $h_i(d)$ for the i -th ego plant defined as the infection probability $\mathbb{P}(\cdot)$ of the ego plant on day d given the data in the hourly environmental data, the daily ego plant features and the neighbor plants' features and distances. Furthermore, an infection on day $d = d_{\text{inf}}^{(i)}$ implies that the plant stayed healthy on all other days before. So formally, it is

$$\begin{aligned} h_i(d) &:= \mathbb{P} \left(d = d_{\text{inf}}^{(i)} \mid d_{\text{inf}}^{(i)} \geq d, \mathbf{E}_i^{\text{hour}}(d'), \mathbf{P}_{\text{ego},i}(d'), \{\mathbf{P}_{\text{nbr},i}^k(d'), \vec{x}_{i,k}\}_{k=1}^K, d' \in \mathcal{D} \right) \\ &= \mathbb{P} \left(d = d_{\text{inf}}^{(i)} \mid d_{\text{inf}}^{(i)} \geq d, \mathcal{H}_{id} \right), \end{aligned} \quad (4.6)$$

where $\mathbf{E}_i^{\text{hour}}(d) \in \mathbb{R}^{24 \times E_{\text{env}}}$ is the hourly environmental data of E_{env} measurands, $\mathbf{P}_{\text{ego},i}(d) \in \mathbb{R}^{P_{\text{ego}}}$ are the daily plant-specific parameters of the ego plant from P_{ego} features, and $\mathbf{P}_{\text{nbr},i}^k(d) \in \mathbb{R}^{P_{\text{nbr}}}$ are the daily plant-specific parameters of the k -th neighbor plant with P_{nbr} features at day d . The vector $\vec{x}_{i,k}$ represents the relative position vector of the k -th neighbor plant with respect to the i -th ego plant. The history $\mathcal{H}_{id}$ summarizes all those data tensors. Furthermore, the data tensors are defined as $\mathbf{E}^{\text{hour}} \in \mathbb{R}^{D \times 24 \times E_{\text{env}}}$, $\mathbf{P}_{\text{ego}} \in \mathbb{R}^{D \times P_{\text{ego}}}$, and $\mathbf{P}_{\text{nbr}} \in \mathbb{R}^{D \times K \times P'_{\text{nbr}}}$, respectively while omitting the ego plant index i . Note that the relative position vectors are concatenated with the neighbor plant features, i.e., $P'_{\text{nbr}} := P_{\text{nbr}} + \dim(\vec{x}_k)$.

Given Rules R6 and R7 in particular, one can derive more details about the infection process. Since the infection event is a Bernoulli process, the probability of an infection on day $d = d_{\text{inf}}^{(i)}$ is expressible by the hazard $h_i(d)$ and the survival function $S_i(d)$, which is the probability that the plant stayed healthy until day d . Thus, it is

$$S_i(d) := \mathbb{P} \left(d < d_{\text{inf}}^{(i)} \mid \mathcal{H}_{id} \right) = \prod_{d'=1}^{d-1} (1 - h_i(d')). \quad (4.7)$$

Using the survival function yields the probability $\mathbb{P}(\cdot)$ of an infection event at day $d = d_{\text{inf}}^{(i)}$ as

$$\mathbb{P}(d = d_{\text{inf}}^{(i)} | \mathcal{H}_{id}) = h_i(d) S_i(d). \quad (4.8)$$

4.7 SpreadNet

In the following, the function $h_i(d)$ —defined in Equation (4.6)—is aimed to be approximated by a neural network (NN) trained with the gathered infection data. This model is called *SpreadNet* in the following. Having hourly weather data is very convenient for the detailed analysis of infection potential by having insights into the day-night-cycle directly available. Moreover, previous CLS spread modeling approaches have shown that hourly or intra-daily information is in fact beneficial for the prediction [124]. However, one does not know per se how different weather parameters at different points of day-time influence or correlate to each other. Thus, the processing of the time series data is based on the transformer architecture [144]. With its attention mechanism (cf. Section 1.2), it is able to correlate parameters across all daily hours without any convolution that is limited in its field of view like in CNNs. Additionally, the transformer architecture is able to process variable-length sequences, which is beneficial for the temporal window of D days. By calculation of the attention maps—as already used in Section 3.5.6—one can interpret the model’s focus on certain parameters at certain times. This provides insights into the infection process itself by mainly data-driven analysis.

Following sections will describe data preprocessing, the architecture of SpreadNet, and the training and evaluation procedure.

4.7.1 Data Preprocessing and Augmentation

Before putting them into SpreadNet, the data tensors need to be properly normalized—or standardized—so that each parameter is inside a reasonable numeric range around zero. For the distance vectors $\vec{x}_k$, one could naively think of a similar scaling approach. However, providing this information directly in Cartesian coordinates $\vec{x}_k = [x_k - x_{\text{ego}}, y_k - y_{\text{ego}}, z_k - z_{\text{ego}}]^\top = [\Delta x_k, \Delta y_k, \Delta z_k]^\top$ is not ideal. From epidemiological modeling it is intuitive that the distance between plants is a crucial factor for the infection probability. However, the *absolute* direction of the distance vector should be less important. It is rather important which *relative* direction, for instance, the direction of the neighbor plant and the wind direction have. Since there is no prior knowledge about any anisotropy of CLS spread dynamics, the model should not get any inductive bias [46] from the coordinate system. Hence, the coordinates are not split into longitude

and latitude but rather into distance and direction by performing a coordinate transform. Thus, cylindrical coordinates $(r, \varphi, \Delta z)$ are introduced with

$$r_k = \sqrt{\Delta x_k^2 + \Delta y_k^2}, \quad \varphi_k = \arctan2(\Delta x_k, \Delta y_k), \quad \Delta z_k = \Delta z_k, \quad (4.9)$$

where the $\arctan2$ function is the generalized arctan function that accounts for the definition gaps. The angle itself, however, is an inappropriate coordinate for processing in NNs due to its periodical discontinuity. For instance, the distance between 359° and 1° seems to be much larger than between 0° and 2° if considering pure numerics. Thus, the angle φ is split into its trigonometrical functions on the unit circle, namely $\sin \varphi$ and $\cos \varphi$. Both are required to have an at least injective map. Since the angles are given as $\varphi = \varphi \bmod 2\pi$, the map

$$\Phi : [0, 2\pi) \rightarrow [-1, 1]^2 ; \quad \varphi \mapsto (\sin \varphi, \cos \varphi) \quad (4.10)$$

is bijective. As a favorable side effect, the results of this mapping are intrinsically normalized due to the codomain being the unit circle. Finally, it is intuitive to bias the model to the fact that nearby plants are more important than remote ones. Moreover, it might be beneficial to have a higher numerical resolution for those nearby neighbors. Thus, the distance data r is inverted ending up in the following representation for the neighbor distance vectors given by

$$\vec{x}_k := [r_k^{-1}, \sin \varphi_k, \cos \varphi_k, \Delta z_k]^\top. \quad (4.11)$$

Note that the transformation Φ is also applied to other directive parameters such as, for instance, the wind direction for the same reasons.

Especially in this case, where only a limited amount of data from only a few field experiments in one geographic region is available, data augmentation is crucial to avoid overfitting and to improve the model's generalization capabilities. The following augmentation strategies are applied to the data tensors with a probability of p_{act} for each augmentation type. The parameters for each augmentation are summarized in Table 4.2.

Random Environment Rotation In order to avoid any bias from the coordinate system and the model's dependency on absolute directions, the entire environment including all position vectors and directional parameters is rotated around the vertical axis by a random angle. Since only limited data from a limited geographic region is given, this augmentation supports the model learning direction-invariant features.

Random Neighbor Dropout Neighbor plants in $\mathcal{H}_{id}$ are randomly dropped with a specific dropout probability. This way, the model cannot rely on specific neighbors all the time and has to learn to generalize over different neighbor constellations.

Random Data Dropout Sensors can fail or provide faulty data. To simulate this behavior and to make the model robust against missing data, single measurements—but not complete series—are randomly dropped with a certain probability. This is done for environmental and plant features independently. Moreover, either single hours or days or the complete feature vector of a feature are randomly dropped.

Random Coordinate Noise The plant positions are not perfectly known due to inaccuracies in finding the exact plant centroid in the cataloging workflow (cf. Chapter 2). To simulate this uncertainty, random coordinate noise is added to the position vectors of the neighbor plants by sampling from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ with zero mean and a certain standard deviation (std) for each coordinate axis σ_x , σ_y , and σ_z . This noise only applied to a fraction p_{noise} of the neighbor plants in each data sample.

Random Sensor Noise Sensor measurements are prone to noise as well. Thus, random sensor noise is added to the measurements of those sensors whose systematic or stochastic measurement errors are given in the datasheets or specifications. In this case, Gaussian noise is added to the temperature, relative humidity, wind speed, and wind direction measurements with a probability of p_{noise} . Note that if after adding noise the values exceed physical limits, e.g., relative humidity $< 0\%$ or $> 100\%$, they are clipped to the valid range.

Random Plant Parameter Noise The plant parameters are directly or indirectly calculated from sensor measurements and thus prone to errors as well. Random noise is added to those parameters based on estimated measurement uncertainties. Similar to the sensor noise, Gaussian noise is

Random Environment Rotation	
p_{act}	0.75
Random Neighbor Dropout	
p_{act}	0.05
dropout	0.1
Random Data Dropout	
p_{act}	0.02
Env. feature dropout	0.02
Env. hour dropout	0.04
Plant feature dropout	0.02
Plant day dropout	0.04
Random Coordinate Noise	
p_{act}	0.05
p_{noise}	0.3
Noise distribution	$\mathcal{N}(0, \sigma^2)$
σ_x, σ_y	0.01 m
σ_z	0.001 m
Random Sensor Noise	
p_{act}	0.02
p_{noise}	0.1
Noise distribution	$\mathcal{N}(0, \sigma^2)$
$\sigma_{\text{temperature}}$	0.5 °C
$\sigma_{\text{relative humidity}}$	5 %
$\sigma_{\text{wind speed}}$	0.3 m/s
$\sigma_{\text{wind direction}}$	3°
Random Plant Parameter Noise	
p_{act}	0.02
p_{noise}	0.1
Noise distribution	$\mathcal{N}(0, \sigma^2)$
σ_{BBCH}	0.5
σ_c	0.01
σ_{ETo}	1 mm/d
σ_{GDD}	3 °C d
σ_r	0.2 cm
Random Infection Day Jitter	
p_{act}	0.05
Δ_{max}	± 3 d
Shift distribution	$\mathcal{N}(0, \sigma^2)$
σ	1.5 d

Table 4.2: Data augmentation parameters for SpreadNet. Summary of all augmentation parameters for training and validation of SpreadNet.

added to the plant parameters with a probability of p_{noise} and they are clipped to valid ranges if necessary.

Random Infection Day Jitter As the infection day is retrieved from visual disease severity annotations combined with the backpropagation of the incubation cycle (cf. Section 4.3), there surely is some uncertainty in this date. To simulate this uncertainty, the infection day is randomly jittered by a discretized Gaussian noise around the retrieved day. As an upper bound, a maximum shift Δ_{max} is set to avoid too large deviations.

All of the above augmentation strategies are applied with a certain activation probability p_{act} . The specific parameters for each augmentation strategy used in this chapter are summarized in Table 4.2. All the augmentations are applied independently during training. For validation, only the random environment rotation and the random infection day jitter are used. Using augmentation during validation is rather untypical. However, since the amount of data is limited and the generalization capabilities of SpreadNet are important, these two augmentations are applied in order to at least demonstrate the model’s rotation invariance. Moreover, the exact infection day for the validation data is a rather uncertain label, the robustness against small shifts in the infection day has to be validated as well.

In the next section, the SpreadNet architecture and its design choices are discussed in detail.

4.7.2 SpreadNet Architecture

In the following, the architecture of SpreadNet is concretized in detail. Figure 4.7 gives an overview of the whole architecture. SpreadNet implements the hazard function $h_i(d)$ defined in Equation (4.6) as a transformer-based NN that processes a spatio-temporal window. It takes three input windows in reversed time (current day d and $D - 1$ past days, cf. Rule R2):

- hourly environmental data $\mathbf{E}^{\text{hour}} \in \mathbb{R}^{24D \times E_{\text{env}}}$ for the latest D days,
- daily ego-plant features $\mathbf{P}_{\text{ego}} \in \mathbb{R}^{D \times P_{\text{ego}}}$,
- daily neighbor features (including relative coordinates) $\mathbf{P}_{\text{nbr}} \in \mathbb{R}^{D \times K \times P'_{\text{nbr}}}$,
- a daily validity mask $\mathbf{M}_{\text{day}} \in \{0, 1\}^D$,
- a neighbor validity mask $\mathbf{M}_{\text{nbr}} \in \{0, 1\}^{D \times K}$, and
- an absolute day index sequence $\mathbf{a}_{\text{abs}} \in \mathbb{N}_0^D$ containing the absolute calendar day for each position in the window.

The validity masks are necessary for days outside the time range from sowing to infection and for padding in a batch-wise processing setup. With these inputs, SpreadNet performs the following steps.

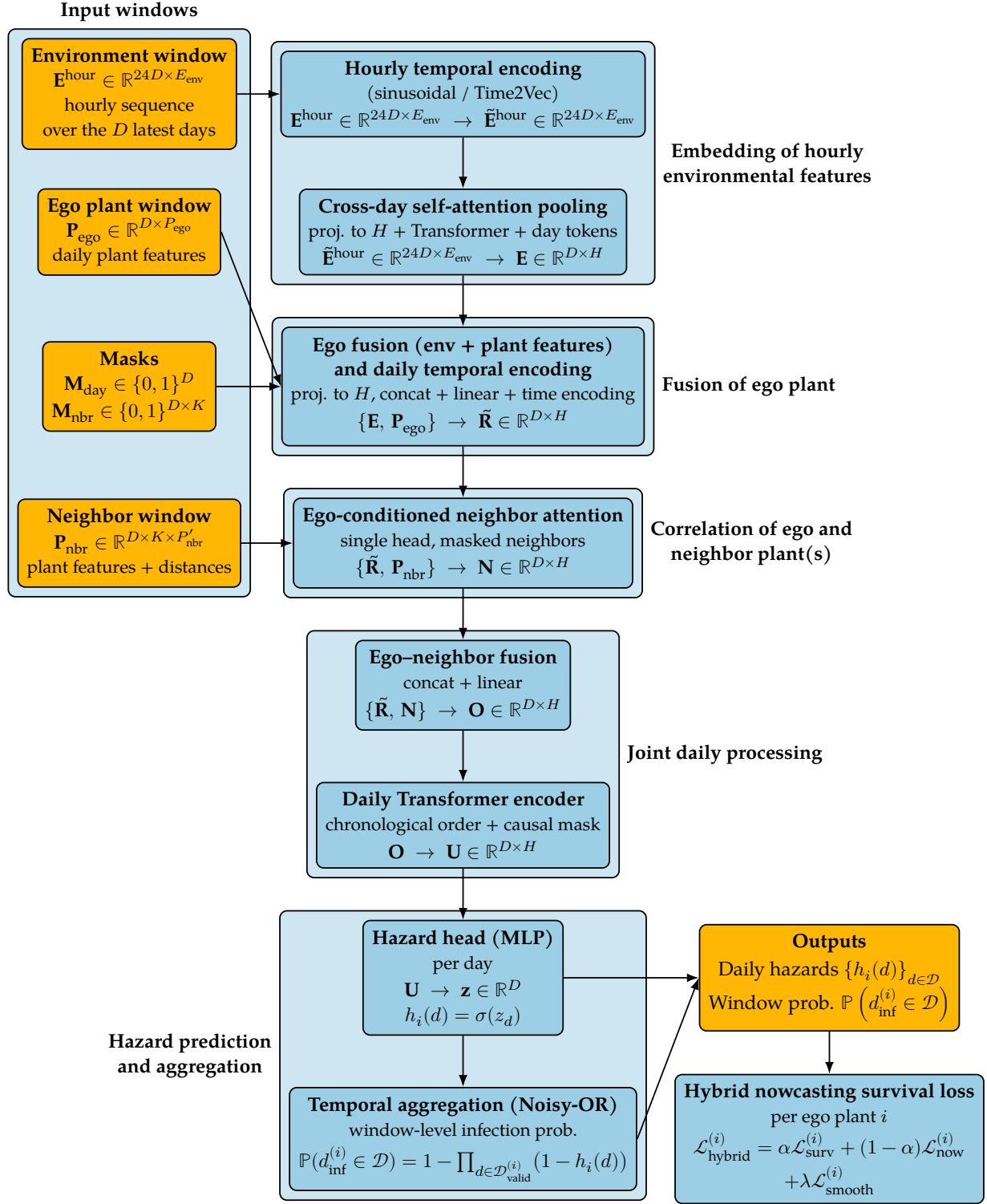


Figure 4.7: Overview of the SpreadNet architecture. The model takes a spatio-temporal window of hourly environment, ego-plant and neighbor features, plus validity masks and absolute day indices. The hourly environment is temporally encoded, processed by a cross-day self-attention block with per-day CLS tokens, and pooled to daily environment embeddings. These are fused with ego-plant features and daily time encodings and enriched with ego-conditioned neighbor information. A causal daily Transformer models temporal dynamics and yields per-day hidden states, which are mapped to daily logits z_d and hazards $h_i(d)$. A Noisy-OR aggregation produces a window-level infection probability, and training is performed with a hybrid nowcasting survival loss that combines survival NLL, current-day nowcasting, and temporal smoothness.

4.7.2.1 Hourly Temporal Encoding

SpreadNet should learn hourly environmental patterns that influence the infection process without being biased by the absolute hour of the day. Thus, the hourly data is treated as a single sequence across all D days in the window. Thus, let $L = 24D$ and let $l \in \{1, \dots, L\}$ be the flattened sequence index in reversed-time order as before. The environmental vector defined by $\mathbf{e}_l \in \mathbb{R}^{E_{\text{env}}}$ can be collected into

$$\mathbf{E}^{\text{hour}} := (\mathbf{e}_l)_{l=1}^L \in \mathbb{R}^{L \times E_{\text{env}}} . \quad (4.12)$$

A temporal encoding ϕ_l is then added along this flattened hourly axis. ϕ_{hour} is either a sinusoidal positional encoding as proposed in [144] or a learnable *Time2Vec* encoding as proposed in [70]. For each position l , one obtains

$$\tilde{\mathbf{e}}_l := \mathbf{e}_l + \phi_{\text{hour}}(l) , \quad (4.13)$$

yielding an encoded hourly sequence $\tilde{\mathbf{E}}^{\text{hour}} \in \mathbb{R}^{L \times E_{\text{env}}}$.

4.7.2.2 Cross-Day Self-Attention Pooling

With the temporal encoding added, the time information is now explicitly available to the model. Since all other data—as well as the goal to predict the hazard function—is defined on a daily level, the hourly information needs to be pooled into daily environment embeddings so that they can be combined with the other data. However, a simple averaging or max-pooling would not allow the model to learn complex correlations between different hours of the day or across different days. Thus, the hourly sequence is forwarded to a multi-layer Transformer encoder f_{hourpool} —which is similar to the one in the SugarViT backbone (cf. Section 3.4). With its *self-attention* (cf. Section 1.2.3) each hourly representation of the environmental parameters can *attend* to any hour in the D -day window. [144]

First, the hourly vector is projected into a hidden dimension H via a learnable linear map with weights $\mathbf{W}_{\text{env}} \in \mathbb{R}^{H \times E_{\text{env}}}$ and biases $\mathbf{b}_{\text{env}} \in \mathbb{R}^H$ yielding

$$\mathbf{s}_l := \mathbf{W}_{\text{env}} \tilde{\mathbf{e}}_l + \mathbf{b}_{\text{env}} \in \mathbb{R}^H \quad (4.14)$$

and the sequence $\mathbf{S} \in \mathbb{R}^{L \times H}$. Now, the 24 hourly representations of each day d are pooled into a single daily environment embedding $\mathbf{e}_d \in \mathbb{R}^H$. As in the vision transformer (ViT) architecture, the encoding of a so-called *classification (CLS) token* is used for this purpose. To obtain one pooled representation per day $d \in \mathcal{D}$, a learnable *day token* $\mathbf{c}_d \in \mathbb{R}^H$ is

inserted for each day. Thus, conceptually, the projected sequence is grouped into D daily subsequences of length 24 yielding

$$\mathbf{S}_d := (\mathbf{s}_{d,h})_{h=1}^{24} \in \mathbb{R}^{24 \times H}. \quad (4.15)$$

For each day d , the day token is prepended so that

$$\mathbf{S}_d^+ := (\mathbf{c}_d, \mathbf{s}_{d,1}, \dots, \mathbf{s}_{d,24}) \in \mathbb{R}^{(1+24) \times H}. \quad (4.16)$$

Collecting all days and flattening them back into a single sequence gives

$$\mathbf{S}^+ := (\mathbf{S}_d^+)_{d \in \mathcal{D}} \in \mathbb{R}^{D(1+24) \times H}. \quad (4.17)$$

Finally, the Transformer encoder f_{hourpool} is applied to this extended sequence to perform the attention-based pooling,

$$\hat{\mathbf{S}}^+ := f_{\text{hourpool}}(\mathbf{S}^+), \quad (4.18)$$

and its output is reshaped back into day-wise subsequences

$$\hat{\mathbf{S}}_d^+ := (\hat{\mathbf{c}}_d, \hat{\mathbf{s}}_{d,1}, \dots, \hat{\mathbf{s}}_{d,24}) \in \mathbb{R}^{(1+24) \times H}. \quad (4.19)$$

The transformed day token $\hat{\mathbf{c}}_d$ serves as the daily environment embedding for day d , i.e.,

$$\mathbf{e}_d := \hat{\mathbf{c}}_d \in \mathbb{R}^H. \quad (4.20)$$

Hence, one obtains the daily pooled sequence

$$\mathbf{E} := (\mathbf{e}_d)_{d \in \mathcal{D}} \in \mathbb{R}^{D \times H}. \quad (4.21)$$

4.7.2.3 Fusion of Ego-Plant Features and Daily Time Encoding

From now on, the model operates on a daily time scale and the daily environment embeddings are fused with the ego-plant features. For each day $d \in \mathcal{D}$, let $\mathbf{e}_d \in \mathbb{R}^H$ denote the pooled environment embedding from the previous step and $\mathbf{p}_d^{\text{ego}} \in \mathbb{R}^{P_{\text{ego}}}$ the ego-plant feature vector. Firstly, the ego features are projected into the same hidden dimension H , again via a learnable linear map with weights $\mathbf{W}_{\text{ego}} \in \mathbb{R}^{H \times P_{\text{ego}}}$ and biases $\mathbf{b}_{\text{ego}} \in \mathbb{R}^H$,

$$\hat{\mathbf{p}}_d^{\text{ego}} := \mathbf{W}_{\text{ego}} \mathbf{p}_d^{\text{ego}} + \mathbf{b}_{\text{ego}} \in \mathbb{R}^H. \quad (4.22)$$

Secondly, the two representations are concatenated to form the raw fused vector

$$\mathbf{r}_d^{\text{raw}} := \begin{bmatrix} \mathbf{e}_d \\ \hat{\mathbf{p}}_d^{\text{ego}} \end{bmatrix} \in \mathbb{R}^{2H}. \quad (4.23)$$

A linear fusion layer with learnable weights $\mathbf{W}_{\text{fuse-ego}} \in \mathbb{R}^{H \times 2H}$ and biases $\mathbf{b}_{\text{fuse-ego}} \in \mathbb{R}^H$ maps this concatenation back into the hidden space,

$$\mathbf{r}_d := \mathbf{W}_{\text{fuse-ego}} \mathbf{r}_d^{\text{raw}} + \mathbf{b}_{\text{fuse-ego}} \in \mathbb{R}^H, \quad (4.24)$$

yielding the daily sequence

$$\mathbf{R} := (\mathbf{r}_d)_{d \in \mathcal{D}} \in \mathbb{R}^{D \times H}. \quad (4.25)$$

Temporal information on the daily level is preserved by adding a daily time encoding to each $\mathbf{r}_d$. Again, either sinusoidal or Time2Vec encoding $\psi_{\text{day}}(d) \in \mathbb{R}^H$ is used. Hence,

$$\begin{aligned} \tilde{\mathbf{r}}_d &:= \mathbf{r}_d + \psi_{\text{day}}(d), \\ \tilde{\mathbf{R}} &:= (\tilde{\mathbf{r}}_d)_{d \in \mathcal{D}} \in \mathbb{R}^{D \times H}, \end{aligned} \quad (4.26)$$

where $\tilde{\mathbf{R}}$ is the final ego-day embedding sequence.

4.7.2.4 Ego-conditioned Neighbor Attention

Now that daily embeddings for the ego plant and environment are obtained, it is desirable to join the spatial neighborhood information as well by introducing an attention mechanism that is conditioned on the ego plant embedding and attends over the neighbor plant embeddings. For each day $d \in \mathcal{D}$, there are K neighbors with feature vectors $\mathbf{p}_{d,k}^{\text{nbr}} \in \mathbb{R}^{P'_{\text{nbr}}}$. SpreadNet applies a single-head ego-conditioned attention to aggregate neighbor information into a fixed-size embedding per day. First, the ego-day features $\tilde{\mathbf{r}}_d$ and neighbor features are linearly projected to queries, keys, and values with learnable weight matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$, yielding matching dimensions:

$$\begin{aligned} \mathbf{q}_d &:= \mathbf{W}_Q \tilde{\mathbf{r}}_d \in \mathbb{R}^H, \\ \mathbf{k}_{d,k} &:= \mathbf{W}_K \mathbf{p}_{d,k}^{\text{nbr}} \in \mathbb{R}^H, \\ \mathbf{v}_{d,k} &:= \mathbf{W}_V \mathbf{p}_{d,k}^{\text{nbr}} \in \mathbb{R}^H. \end{aligned} \quad (4.27)$$

This model part should combine ego and neighbor features by focusing on their patterns rather than their magnitudes. Thus, both queries and keys are ℓ_2 -normalized,

$$\hat{\mathbf{q}}_d := \frac{\mathbf{q}_d}{\|\mathbf{q}_d\|_2}, \quad \hat{\mathbf{k}}_{d,k} := \frac{\mathbf{k}_{d,k}}{\|\mathbf{k}_{d,k}\|_2}, \quad (4.28)$$

so that the dot-product yields the attention scores

$$a_{d,k} := \hat{\mathbf{q}}_d^\top \hat{\mathbf{k}}_{d,k} \quad (4.29)$$

as cosine similarities between ego and neighbor embeddings. After masking invalid neighbors, a numerically safe application of the softmax function yields the attention weights

$$\beta_{d,k} := \frac{\exp(a_{d,k})}{\sum_{k' \in \mathcal{K}_d} \exp(a_{d,k'})}, \quad (4.30)$$

where $\mathcal{K}_d$ is the set of valid neighbors on day d . The attention weights $\beta_{d,k}$ are now directly interpretable as the importance of neighbor k to the ego plant on day d . During training, they can be randomly put to zero to encourage robustness by attention dropout. With those weights, a neighbor aggregation for day d is computed as

$$\mathbf{n}'_d := \sum_{k \in \mathcal{K}_d} \beta_{d,k} \mathbf{v}_{d,k} \in \mathbb{R}^H, \quad (4.31)$$

In the next step, a learned output projection,

$$\mathbf{n}_d := \mathbf{W}_O \mathbf{n}'_d + \mathbf{b}_O \in \mathbb{R}^H, \quad (4.32)$$

is applied with learnable weights $\mathbf{W}_O \in \mathbb{R}^{H \times H}$ and biases $\mathbf{b}_O \in \mathbb{R}^H$. This matrix $\mathbf{W}_O$ plays an analogous role to the output projection in standard attention mechanisms [144]. It yields the final ego-conditioned neighbor embedding sequence. The ego-conditioning in this step allows the model to learn the neighbor states as a function of the ego plant's condition and environment. Without the ego-conditioning—i.e., by applying self-attention over the neighbors only—the model could not learn to focus on neighbors' states that are relevant for the ego plant's particular situation, but it could only learn a general summary of the neighbors' conditions. Thus, the ego-conditioned neighbor embedding sequence is

$$\mathbf{N} := (\mathbf{n}_d)_{d \in \mathcal{D}} \in \mathbb{R}^{D \times H}. \quad (4.33)$$

4.7.2.5 Ego-Neighbor Fusion

After computing the ego-day embedding $\tilde{\mathbf{r}}_d \in \mathbb{R}^H$ and the aggregated neighbor embedding $\mathbf{n}_d \in \mathbb{R}^H$, these two representations are fused by a single linear layer. First, the concatenated vector is built as

$$\mathbf{o}_d^{\text{raw}} := \begin{bmatrix} \tilde{\mathbf{r}}_d \\ \mathbf{n}_d \end{bmatrix} \in \mathbb{R}^{2H}. \quad (4.34)$$

The fusion layer then applies a learned linear mapping with weights $\mathbf{W}_{\text{fuse-nei}} \in \mathbb{R}^{H \times 2H}$ and biases $\mathbf{b}_{\text{fuse-nei}} \in \mathbb{R}^H$ which yields

$$\mathbf{o}_d := \mathbf{W}_{\text{fuse-nei}} \mathbf{o}_d^{\text{raw}} + \mathbf{b}_{\text{fuse-nei}} \in \mathbb{R}^H. \quad (4.35)$$

For interpretability, $\mathbf{W}_{\text{fuse-nei}}$ can be decomposed into two blocks acting on the ego and neighbor parts separately:

$$\mathbf{W}_{\text{fuse-nei}} := [\mathbf{W}_{\text{self}} \quad \mathbf{W}_{\text{nei}}], \quad \mathbf{W}_{\text{self}}, \mathbf{W}_{\text{nei}} \in \mathbb{R}^{H \times H}, \quad (4.36)$$

so that

$$\mathbf{o}_d := \mathbf{W}_{\text{self}} \tilde{\mathbf{r}}_d + \mathbf{W}_{\text{nei}} \mathbf{m}_d + \mathbf{b}_{\text{fuse-nei}}. \quad (4.37)$$

This makes the role of the fusion layer more explicit. In particular, $\mathbf{W}_{\text{self}}$ learns how to transform the ego-plant context into the representation space used by the temporal transformer, while $\mathbf{W}_{\text{nei}}$ controls how the aggregated neighbor information $\mathbf{m}_d$ influences the per-day context. For example, if the matrix elements in $\mathbf{W}_{\text{nei}}$ are close to zero, the model effectively ignores neighbors; if $\mathbf{W}_{\text{nei}}$ has a large norm, the ego context is strongly influenced by the condition and spatial configuration of neighboring plants. Moreover, the bias $\mathbf{b}_{\text{fuse-nei}}$ allows the model to shift the fused representation independently of the inputs, for instance to represent an underlying infection pressure even in the absence of infected neighbors. Finally, the fused representation $\mathbf{o}_d$ is the final per-day context vector. In tensor form, stacking over days yields

$$\mathbf{O} := (\mathbf{o}_d)_{d \in \mathcal{D}} \in \mathbb{R}^{D \times H}, \quad (4.38)$$

which is then processed by the next model part.

4.7.2.6 Daily Transformer Encoder

The sequence $\mathbf{O}$ is currently ordered in reversed time, i.e. index 0 corresponds to day d , index 1 to $d - 1$, etc. For temporal modeling using a causal transformer encoder architecture as in [144] out of the box, the sequence has to be reordered into chronological order so that day $d - D + 1$ comes first and day d last. This way, a causal attention mask can be applied to ensure that day d can only attend to itself and earlier days, but not to future days. Thus, the valid part of each sequence is reordered into chronological order, the causal mask is applied, and finally reversed back. Let μ_i denote the number of valid days for plant i in the window obtained from $\mathbf{M}_{\text{day}}$. For all plants with the same $\mu = \mu_i$, the model flips the first μ entries of $\mathbf{O}$:

$$\mathbf{O}_{i,1:\mu}^{\text{chron}} := (\mathbf{o}_{i,d-L\mu+1}, \dots, \mathbf{o}_{i,d-1}, \mathbf{o}_{i,d}) . \quad (4.39)$$

A causal attention mask $\mathbf{M}^{\text{causal}} \in \{0, 1\}^{\mu \times \mu}$ enforces that day j can only attend to itself and earlier days:

$$M_{jk}^{\text{causal}} := \begin{cases} 0, & k \leq j \quad (\text{allowed}), \\ 1, & k > j \quad (\text{masked}). \end{cases} \quad (4.40)$$

The chronological daily sequence is then processed by the transformer encoder with multi-head self-attention layers and position-wise feed-forward networks as in [144] (cf. Section 1.2.4). In principle, the same network that has been used for the SugarViT model backbone (cf. Section 3.4) is used here. Hence, the transformer encoder yields the output sequence

$$\mathbf{U}_{i,1:\mu}^{\text{chron}} := \text{Transformer}(\mathbf{O}_{i,1:\mu}^{\text{chron}}, \mathbf{M}^{\text{causal}}) \quad (4.41)$$

and the resulting sequence $\mathbf{U}^{\text{chron}}$ which is flipped back into reversed order to obtain $\mathbf{U} \in \mathbb{R}^{D \times H}$ aligned with the original indexing scheme.

4.7.2.7 Hazard Head and Daily Infection Hazards

A small multi-layer perceptron (MLP) classifier maps each daily hidden state $\mathbf{u}_d \in \mathbb{R}^H$ to a scalar logit $z_d \in \mathbb{R}$:

$$z_d := \text{MLP}(\mathbf{u}_d) . \quad (4.42)$$

A logit is defined as the logarithm of the odds of an event occurring. Here, it represents the unnormalized score for the infection hazard on day d . Hence,

$$z_d = \text{logit}(h_i(d)) := \log \left(\frac{h_i(d)}{1 - h_i(d)} \right). \quad (4.43)$$

Invalid (padded) days are forced to a very low logit so that the corresponding hazard is essentially zero. The inverse of the logit function is the sigmoid function defined as

$$\sigma(x) := \frac{1}{1 + \exp(-x)}. \quad (4.44)$$

Using it as an activation function, one transfers the logits into probabilities. Hence, it yields the daily infection hazard as

$$h_i(d) = \sigma(z_d) = \frac{1}{1 + \exp(-z_d)}, \quad (4.45)$$

which is in line with the definition of $h_i(t)$ in Equation (4.6). Collecting all valid days, it is

$$\mathbf{z} := (z_d)_{d \in \mathcal{D}} \in \mathbb{R}^D, \quad \mathbf{h} := (h_i(d))_{d \in \mathcal{D}} \in (0, 1)^D. \quad (4.46)$$

Temporal Aggregation via Noisy-OR Besides day-wise hazards $h_i(d)$, one may also require the probability that an infection occurs at least once within the window of D days. Assuming conditional independence of daily infection events given the inputs, the probability of *no* infection in the window is the product of daily survival probabilities. Hence, analogous to Equation (4.7), the probability of survival over the window of valid days $\mathcal{D}_{\text{valid}}$ for plant i is

$$S_i(\mathcal{D}) := \prod_{d \in (\mathcal{D}_{\text{valid}}^{(i)} \cap \mathcal{D})} (1 - h_i(d)). \quad (4.47)$$

The window-level infection probability is then modeled as a so-called *Noisy-OR* [111] aggregation of daily hazards which yields

$$\mathbb{P}(d_{\text{inf}}^{(i)} \in \mathcal{D}) := 1 - S_i(\mathcal{D}) = 1 - \prod_{d \in (\mathcal{D}_{\text{valid}}^{(i)} \cap \mathcal{D})} (1 - h_i(d)). \quad (4.48)$$

Now that the full SpreadNet architecture is defined, its training procedure is outlined.

4.7.3 Hybrid Nowcasting Survival Loss Function

As the initial step, the loss function is defined. Training SpreadNet requires a loss that is consistent with the discrete-time survival formulation from Equation (4.7), but that also emphasizes accurate nowcasting (cf. Rule R4) of infections on the current day d . Three components are combined to form a hybrid loss function that meets these requirements.

Survival NLL Let $z_i(d)$ be the daily logit for plant i and day index d , and $h_i(d) = \sigma(z_i(d))$. The discrete-time survival NLL [57, 137] for ego plant i can be written as

$$\mathcal{L}_{\text{surv}}^{(i)} := \begin{cases} -\log h_i(d) - \sum_{d=1}^{d_{\text{inf}}^{(i)}-1} \log(1 - h_i(d)) , & d = d_{\text{inf}}^{(i)} , \\ -\sum_{d=1}^d \log(1 - h_i(d)) , & d < d_{\text{inf}}^{(i)} , \end{cases} \quad (4.49)$$

following the definitions in Rule R2 and R3. The days $d > d_{\text{inf}}^{(i)}$ are censored and not considered in the model, and, thus, can be ignored in the loss function. In practice, the loss is not evaluated over the complete history from day $d = 1$ onward, but only over a finite time window as defined in Rule R2. This windowed evaluation remains a valid training objective under the discrete-time survival formulation. With the windowed approach, the loss becomes a conditional likelihood that accounts only for the days within the window. However, since the model assumes a Markovian structure of daily hazards, this restriction does not affect the probabilistic interpretation of the loss. The “survival factor” saying that the plant has survived until the earliest day in the window cannot be influenced by the model parameters and therefore cancels out in the conditional likelihood; with other words, if the plant had been infected before the window started, one would not use it as a data point due to Rule R3.

Nowcasting BCE Now, the loss respects the survival aspect of hazard modeling. However, since the accurate prediction especially on the current day d (cf. Rule R4) is of major interest, an auxiliary nowcasting loss term is added that directly optimizes the binary classification of whether the infection occurs on day d or not. First, Let

$$y^{(i)}(d) := \begin{cases} 1, & d = d_{\text{inf}}^{(i)} , \\ 0, & d < d_{\text{inf}}^{(i)} \end{cases} \quad (4.50)$$

be a binary nowcasting label for the current day d . With that, the BCE loss for the current-day hazard $h_i(d)$ is

$$\mathcal{L}_{\text{now}}^{(i)} := -y^{(i)}(d) \log h_i(d) - (1 - y^{(i)}(d)) \log(1 - h_i(d)) . \quad (4.51)$$

Smoothness Regularization The infection event is based on environmental conditions that typically evolve smoothly over time. Abrupt changes in the estimated daily hazard from one day to the next are therefore unlikely. To incorporate this domain knowledge into the model, a smoothness regularization term is introduced in order to encourage smooth temporal evolution of the daily hazards within a window. The regularization is modeled as an ℓ_1 -based penalty for the daily logits over the valid days which yields

$$\mathcal{L}_{\text{smooth}}^{(i)} := \frac{1}{D-1} \sum_{d'=d-D+2}^d |z_i(d') - z_i(d'-1)|. \quad (4.52)$$

The above three components are combined into a hybrid loss named the *hybrid now-casting survival loss*, which is a weighted sum of the survival NLL and the nowcasting BCE penalized by the smoothness term. Hence,

$$\mathcal{L}_{\text{hybrid}}^{(i)} := \alpha \mathcal{L}_{\text{surv}}^{(i)} + (1 - \alpha) \mathcal{L}_{\text{now}}^{(i)} + \lambda \mathcal{L}_{\text{smooth}}^{(i)}, \quad (4.53)$$

with the weighting hyperparameter $\alpha \in [0, 1]$ and the penalty hyperparameter $\lambda \geq 0$. In the following experiments, the hyperparameters are set to specific $\alpha = 0.3$ and $\lambda = 1 \times 10^{-3}$. The batch loss is obtained by averaging $\mathcal{L}_{\text{hybrid}}^{(i)}$ over all plants in the batch with at least one valid day.

4.7.4 Sampling Strategy

In the modeling scenario, each single plant can only have at most one single infection event during the season (cf. Rule R3). Therefore, if random plants and random windows $\mathcal{D}_d^{(i)} \subset \mathcal{D}_{\text{valid}}^{(i)}$ are sampled, the large majority of them are negative samples—i.e., no infection in the window—leading to a strong class imbalance. A positive sample for the i -th plant is defined as a window in that the infection day $d_{\text{inf}}^{(i)}$ is the latest day and a negative sample as a window that does not include the infection day. More formally,

$$\mathcal{D}_{\mp}^{(i)} := \left\{ \mathcal{D}_d \mid d \leq d_{\text{inf}}^{(i)} \wedge |\mathcal{D}_d| = \tilde{D} \right\}, \quad \tilde{D} \sim \mathcal{U}(D_{\text{min}}, D_{\text{max}}). \quad (4.54)$$

During training, the window length $\tilde{D}$ is sampled from a uniform distribution $\mathcal{U}(\cdot)$ between a minimum D_{min} and maximum D_{max} length to increase variability. In the following experiments, $D_{\text{min}} = 3$ and $D_{\text{max}} = 10$ are chosen. Note that the effective window length may still be shorter than $\tilde{D}$ if the sampled window exceeds the valid day range of the plant. Performing this rather straightforward sampling strategy without any balancing between positive and negative samples only leads to a training data distribution that is very similar to the real-world infection distribution. Thus, the model does not need

to deal with any sampling bias or calibration issues at inference time. During validation and testing, however, fixed length windows of size $D_{\text{val/test}}$ are used to evaluate the model performance consistently over all plants and days. In the following experiments, $D_{\text{val/test}} = 7$ is chosen. SpreadNet is trained on a random split of 80 % of the plants in datasets D03, D06, D07, and D11. The remaining 20 % of the plants are used for validation. The dataset D08 is held out for testing. Strictly splitting the training and validation dataset by plants instead of windows ensures that there is no data leakage between training and validation. Otherwise, it happens that, for instance, the model sees data from day $d - 1$ of plant i during training and then is evaluated on day d of the same plant during validation, which would lead to overly optimistic performance estimates.

4.7.5 Evaluation Metrics

During training and evaluation, the performance of SpreadNet is measured especially focusing on its two main objectives:

- O1 The model should accurately nowcast whether an infection occurs on the current day or not.
- O2 The model should accurately predict the infection day for each plant as precisely as possible.

Therefore, two major classes of metrics are used.

4.7.5.1 Nowcasting quality (Objective O1)

Let $\hat{p}_k$ denote the predicted probability of infection on the latest day of the k -th ego-healthy window (i.e., a window that does not contain an infection event yet) in N total predictions. By using the nowcasting labels from Equation (4.50) again, one obtains the ground-truth labels y_k for each window and can evaluate the nowcasting performance with the following metrics.

Brier score (BS) The *Brier score* [21] is simply the mean squared error of the predicted probabilities:

$$\text{BS} := \frac{1}{N} \sum_{k=1}^N (\hat{p}_k - y_k)^2. \quad (4.55)$$

Lower values indicate that the predicted infection probabilities are close to the observed frequencies. It is a good overall measure of prediction accuracy, combining both discrimination and calibration aspects. However, if the class distribution is highly imbalanced, the Brier score may be dominated by the majority class performance.

Brier skill score (BSS) To assess the relative improvement of SpreadNet over a naive baseline, the *Brier skill score* [151] is reported as well. The baseline model is assumed to always predict the base infection rate $\bar{y} = \frac{1}{N} \sum_{k=1}^N y_k$. The BSS is then defined as

$$\text{BSS} := 1 - \frac{\text{BS}}{\text{BS}_{\text{ref}}}, \quad \text{BS}_{\text{ref}} := \frac{1}{N} \sum_{k=1}^N (\bar{y} - y_k)^2. \quad (4.56)$$

A BSS of 1 indicates perfect predictions, a BSS of 0 indicates no improvement over the baseline, and a negative BSS indicates performance worse than the baseline.

AUROC and AUPRC To quantify how well SpreadNet discriminates between days with and without infection, the AUROC and the AUPRC are reported [29]. Since positive infections are rare, the AUPRC, which is more informative under class imbalance is reported. Thus, it better reflects performance on the positive infection class. [92].

Expected calibration error (ECE) To assess the calibration of the nowcasting probabilities, a binning-based ECE is used. Let $[0, 1]$ be partitioned into M disjoint bins $\{B_m\}_{m=1}^M$. For each bin B_m , an average predicted probability (confidence) and the empirical infection frequency (accuracy) is defined as

$$\text{conf}_m := \frac{1}{|B_m|} \sum_{k \in B_m} \hat{p}_k, \quad \text{acc}_m := \frac{1}{|B_m|} \sum_{k \in B_m} y_k. \quad (4.57)$$

The (ℓ_1) -ECE is then

$$\text{ECE} := \sum_{m=1}^M \frac{|B_m|}{N} |\text{acc}_m - \text{conf}_m|. \quad (4.58)$$

In a perfectly calibrated model it is $\text{acc}_m \approx \text{conf}_m$ in every bin, so ECE would be close to zero. Regarding the use case, ECE quantifies whether, for instance, a predicted 10 % risk of infection today corresponds to an actual infection frequency of about 10 % among all (valid) windows.

4.7.5.2 Infection-time prediction (Objective O2)

For this objective, the predictions are recognized as full discrete-time daily hazard over the entire observation period for each plant. Thus, one can evaluate how well the predicted daily hazard captures the actual infection timing. $d_{\text{inf}}^{(i)}$ denotes the true (absolute) day on which plant i becomes infected and $\hat{d}_{\text{inf}}^{(i)}$ its estimate of SpreadNet. Further, let $\delta_i \in \{0, 1\}$ be the event indicator, i.e. $\delta_i = 1$ for observed infection and $\delta_i = 0$ for no

observed infection of plant i in the observation period. The predicted daily hazard trajectory, i.e. the time series of all prediction daily hazards, can be written as $\{\hat{h}_i(d)\}_d$ for plant i . From that, one can derive a discrete-time survival function

$$\hat{S}_i(d) := \hat{\mathbb{P}}(d < d_{\text{inf}}^{(i)} | \mathcal{H}_{id}) = \prod_{d' \leq d} (1 - \hat{h}_i(d')), \quad (4.59)$$

and a discrete-time probability mass function (PMF) for the infection day

$$\hat{p}_i(d = d_{\text{inf}}^{(i)} | \mathcal{H}_{id}) = \hat{h}_i(d) \prod_{d' < d} (1 - \hat{h}_i(d')). \quad (4.60)$$

The following metrics are used to evaluate how well the predicted hazard trajectories from SpreadNet capture the infection timing.

Integrated Brier score (IBS) For each day d , the survival BS is

$$\text{BS}(d) := \frac{1}{N} \sum_{i=1}^N (\hat{S}_i(d) - Y_i(d))^2, \quad Y_i(d) := \chi_{\{d < d_{\text{inf}}^{(i)}\}}. \quad (4.61)$$

Descriptively, it is the mean squared error (MSE) between predicted survival and the indicator function $\chi_{\{\cdot\}}$ of being still healthy at day d . The IBS [48] is calculated by discretely integrating the BS over time. Hence, it is

$$\text{IBS} := \frac{1}{|\mathcal{D}_{\text{tot}}|} \sum_{d \in \mathcal{D}_{\text{tot}}} \text{BS}(d). \quad (4.62)$$

The IBS thus measures the time-dependent quality of the entire hazard trajectory; a lower IBS indicates that the predicted survival curves closely match the empirical survival pattern over time.

Harrell's C-index To quantify whether plants that were infected earlier are, in fact, assigned higher risk, Harrell's concordance index [51] is used. The concordance index—or C-index—measures the ability of a survival model to correctly rank event times. It is defined as the probability that, for a randomly chosen comparable pair of individuals, the one who experiences the event earlier is assigned a higher predicted risk. Hence, for each plant i , one can infer a *risk score* $\hat{r}_i$ from the predicted hazard trajectory. The higher it is, the more likely the plant is to be infected sooner. The i -th plant can be compared to another plant j if at least one of them experienced an infection event (i.e., $\delta_i = 1$ or $\delta_j = 1$) and if the plant with the shorter observed time-to-event is uncensored. Thus, if $d_{\text{inf}}^{(i)} < d_{\text{inf}}^{(j)}$ and $\delta_i = 1$. The C-index is then computed as the proportion of all comparable

pairs in which the plant with the shorter observed time-to-event has a higher risk score. Formally,

$$C := \frac{\sum_i \sum_j \chi_{\{\hat{r}_i > \hat{r}_j\}} \chi_{\{d_{\text{inf}}^{(i)} < d_{\text{inf}}^{(j)}\}} \delta_i}{\sum_i \sum_j \chi_{\{d_{\text{inf}}^{(i)} < d_{\text{inf}}^{(j)}\}} \delta_i}. \quad (4.63)$$

Its value has the following intuitive interpretation:

- $C = 1$: Perfect concordance; all comparable pairs are correctly ranked.
- $C = 0.5$: The model's rankings are completely random.
- $C < 0.5$: Indicates that the model systematically ranks individuals incorrectly or inverted.

The C-index can measure the global ranking performance of the model quite well. However, it is not sensitive to the absolute timing accuracy of the predicted infection days. Thus, it is complemented with the following metrics.

Event-day MAE and RMSE For plants with an observed ($\delta_i = 1$) infection at day $d_{\text{inf}}^{(i)}$, the conditional expected infection day predicted by the model is defined as the expectation value of the predicted day conditioned on an infection event occurring. Hence,

$$\begin{aligned} \tilde{d}_{\text{inf}}^{(i)} &:= \mathbb{E}[d_{\text{inf}}^{(i)} \mid \mathcal{H}_{id}, \delta_i = 1] = \sum_{d \in \mathcal{D}_{\text{tot}}} d \cdot \hat{p}_i(d = d_{\text{inf}}^{(i)} \mid \mathcal{H}_{id}, \delta_i = 1) \\ &= \frac{\sum_{d \in \mathcal{D}_{\text{tot}}} d \cdot \hat{p}_i(d = d_{\text{inf}}^{(i)} \mid \mathcal{H}_{id})}{\sum_{d \in \mathcal{D}_{\text{tot}}} \hat{p}_i(\delta_i = 1 \mid \mathcal{H}_{id})} \stackrel{(4.60)}{=} \frac{\sum_{d \in \mathcal{D}_{\text{tot}}} d \cdot (\hat{h}_i(d) \prod_{d' < d} (1 - \hat{h}_i(d')))}{\sum_{d \in \mathcal{D}_{\text{tot}}} (\hat{h}_i(d) \prod_{d' < d} (1 - \hat{h}_i(d')))}. \end{aligned} \quad (4.64)$$

The event-day MAE and RMSE are then computed as usual between the predicted and true infection days yielding

$$\text{MAE} := \frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} |\tilde{d}_{\text{inf}}^{(i)} - d_{\text{inf}}^{(i)}|, \quad (4.65)$$

$$\text{RMSE} := \sqrt{\frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} (\tilde{d}_{\text{inf}}^{(i)} - d_{\text{inf}}^{(i)})^2}, \quad (4.66)$$

where $\mathcal{J} = \{i \mid \delta_i = 1\}$ is the set of plants with observed infections. These metrics directly express, in units of days, how early or late SpreadNet predicts the infection time, which is particularly intuitive in the given setting.

4.7.6 Model Training

With the defined architecture and loss function, the training procedure of SpreadNet is described stepwise. The modular architecture of SpreadNet allows for very diverse configurations allowing to trade off model complexity and performance for use cases with different requirements on data availability and prediction performance. In order to cover a wide range of configurations, four model sizes are defined: S , M , L , and XL . The model size is measured by the total number of parameters. For all those models, the training procedure is the same. Solely for the XL model, the learning rate is halved to ensure a stable training run. As for SugarViT, AdamW is used as the optimizer. Additionally, a learning rate scheduling with a warmup phase and inverse square root decay as in the original transformer paper [144] is applied. The specific hyperparameters for each model size are listed in Table 4.3.

Parameter	SpreadNet model			
	S	M	L	XL
Dimensions				
E_{ego}	9	9	9	9
P_{ego}	7	7	7	7
P'_{nbr}	9+4	9+4	9+4	9+4
H	128	128	256	512
Hourly temporal encoding (Section 4.7.2.1)				
type	Sinusoidal	Sinusoidal	Time2Vec	Time2Vec
Cross-day self-attention pooling (Section 4.7.2.2)				
# heads	1	1	2	4
# hidden layers	1	2	4	4
intermediate size	512	512	1024	1024
activation hidden layers	ReLU	ReLU	ReLU	ReLU
attention dropout	0.1	0.1	0.1	0.1
Ego-conditioned neighbor attention (Section 4.7.2.4)				
attention dropout	0.05	0.05	0.05	0.05
Daily transformer encoder (Section 4.7.2.6)				
# heads	1	4	8	8
# hidden layers	1	4	8	12
intermediate size	512	512	1024	1024
activation hidden layers	ReLU	ReLU	ReLU	ReLU
attention dropout	0.1	0.1	0.1	0.1
Hazard head (Section 4.7.2.7)				
layer sizes	64 $\rightarrow$ 32	128 $\rightarrow$ 64	128 $\rightarrow$ 64	128 $\rightarrow$ 64
activation	ReLU	ReLU	ReLU	ReLU
normalization	LN	LN	LN	LN
dropout	0.3	0.3	0.3	0.3
Optimizer				
algorithm	AdamW	AdamW	AdamW	AdamW
learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	5×10^{-5}
weight decay	1×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-2}
β	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)
warmup steps	10000	10000	10000	10000
Loss function (Section 4.7.3)				
α	0.3	0.3	0.3	0.3
λ	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Total parameters	509×10^3	1.3×10^6	9.9×10^6	35.2×10^6

Table 4.3: SpreadNet model configurations. To observe the effect of model size on performance, four different model configurations are defined: Small (S), Medium (M), Large (L), and Extra Large (XL). The table summarizes the key hyperparameters and architectural choices for each configuration.

4.7.7 Model Application during Simulation

Once SpreadNet is trained, it can be run side-by-side with the simulation. The simulation is initialized at sowing date (day 0). On each day d , the following steps are performed:

1. For each healthy plant i , the model is queried with all parameters in the current window $\mathcal{D}_d^{(i)}$ to obtain the predicted infection probability $\hat{p}_i(d)$ for the current day.
2. For each healthy plant i , a random number u_i is drawn from a uniform distribution $\mathcal{U}(0, 1)$. If $u_i < \hat{p}_i(d)$, the plant is marked as infected on day d , i.e., $\hat{d}_{\text{inf}}^{(i)} = d$. Otherwise, it remains healthy.
3. The simulation state is updated to reflect the new infections, which may affect the neighbor relationships and the environmental conditions for the next day.

Particularly, the disease-related infection and fructification cycles start to increase from the predicted infection day $\hat{d}_{\text{inf}}^{(i)}$. From this point onward, this infected plant will be considered as a source/neighbor for other healthy plants in the subsequent days, but not as an ego target anymore due to Rule R3. Although the other environment-related parameters are not affected by a plant's infection status, the infection of a plant may lead to changes in the neighbor relationships for other plants. For instance, if plant i becomes infected on day d , it will be considered as a neighbor for other healthy plants in its neighborhood from day $d + 1$ onward. This change in neighbor relationships influences the predicted hazards for those healthy plants in the subsequent days, thus affecting the overall disease propagation dynamics in the simulation. Due to that, the model's predictions and the simulation state are tightly coupled, creating a feedback loop where the model's predictions influence the simulation, and the evolving simulation state influences future predictions. In time series modeling, this is often referred to as an *autoregressive* or *closed-loop* model. This autoregressive interaction allows SpreadNet to capture complex spatio-temporal patterns of disease spread in the field. However, it also means that the model's performance can have a significant impact on the accuracy of the simulated disease spread, making it crucial to ensure that SpreadNet's predictions are accurate. The Bernoulli sampling step furthermore introduces stochasticity into the simulation, which can be beneficial for capturing the inherent randomness in disease spread. The feedback loop connected to this stochastic character can—however—lead to a “butterfly effect”, where small differences in predictions can lead to diverging outcomes in the simulation over time. This again emphasizes the importance to evaluate the model's performance not only on immediate nowcasting accuracy but also on its ability to capture the overall disease spread dynamics over the entire season.

In the next section, the SpreadNet training and evaluation results are presented, followed by a discussion of the findings.

4.8 Results

This section presents the training results of SpreadNet. The performance of the four model sizes (S , M , L , XL) is evaluated on both the validation and test datasets using the defined metrics. The used metrics discussed in Section 4.7.5 help to understand the model’s performance regarding the two main objectives of nowcasting infection events (O1) and predicting infection timing (O2). However, the evaluation solely on the ground truth datasets does not capture the full picture of how well SpreadNet performs in the context of the simulation. As mentioned before in Section 4.7.7, the autoregressive setup and the butterfly effect can lead to diverging outcomes in the simulation over time, even if the model’s immediate nowcasting performance is good. SpreadNet’s prediction on day d directly influences the new “ground truth” data of day $d + 1$.

This is why the following evaluation is structured in two parts. Firstly, the performance of SpreadNet is evaluated on the original datasets using the defined metrics. This provides insights into how well the model can nowcast infection events and predict infection timing based on the observed data. Secondly, SpreadNet is applied side-by-side to the agent-based simulation and the decay of nowcasting quality is analyzed.

4.8.1 Evaluation on the Datasets

The evaluation on the original train/validation and test datasets yields insights into the plant-wise nowcasting and infection-day prediction capabilities of SpreadNet. On the one hand, the metrics for both objectives are evaluated to understand the model’s performance and to compare the different model sizes. In the end, the trade-off between model complexity and performance can be seen by those metrics. On the other hand, the model parts with attention layers involved can be analyzed to understand which temporal patterns and neighbor relationships are most relevant for the model’s predictions. This can provide insights into the underlying disease spread dynamics and help to identify key factors driving the infection process. Furthermore, the relevance of input features can be analyzed by saliency mapping in order to identify more or less important input features.

4.8.1.1 Metrics

During the training process, the metrics proposed in Section 4.7.5 are monitored on the training, validation and test subsets. The best observed value for each metric across the training epochs for each of the four model sizes is reported in Table 4.4. For the validation set, it can be observed that the M model achieves the best performance across nearly all metrics, indicating a good balance between model complexity and generalization. However, on the test set, the larger models L and XL outperform the smaller ones.

This suggests that the larger models may have an even better generalization capacity on unseen data. Especially regarding the event-day MAE and RMSE, the models perform significantly better on the test than on the validation set. The reason might be that the test set (dataset D08) is a rather small set and does not provide as much variability in terms of treatment schemes and spread dynamics than plants in the validation set may have.

Subset	Model	Nowcasting (O1)					Infection-day prediction (O2)			
		BS [10^{-3}]	BSS	AUROC	AUPRC	ECE [10^{-3}]	IBS [10^{-3}]	C-index [10^{-2}]	MAE [d]	RMSE [d]
Training	S	6.16	0.225	0.980	0.364	0.12	—	—	—	—
	M	5.93	0.258	0.983	0.412	0.08	—	—	—	—
	L	5.97	0.245	0.982	0.393	0.11	—	—	—	—
	XL	5.98	0.234	0.980	0.380	0.16	—	—	—	—
Validation	S	6.26	0.213	0.979	0.347	0.16	14.34	90.93	4.21	6.35
	M	6.19	0.221	0.980	0.361	0.12	14.02	91.10	4.12	6.26
	L	6.21	0.219	0.979	0.360	0.17	14.15	91.20	4.15	6.27
	XL	6.25	0.214	0.979	0.351	0.26	14.37	91.17	4.21	6.33
Test	S	7.59	0.149	0.982	0.348	1.02	14.88	94.84	2.54	3.51
	M	7.23	0.189	0.981	0.365	2.46	12.54	94.80	2.58	3.62
	L	6.52	0.269	0.984	0.524	2.00	11.30	94.88	3.11	3.94
	XL	7.71	0.135	0.979	0.368	2.06	14.21	94.90	2.30	3.33

Table 4.4: Performance metrics for SpreadNet variants. The table shows the best observed value for each metric across the training epochs for each of the four model sizes (S, M, L, XL) on training, validation, and test subsets. Bold values indicate the best performance for each metric. The metrics in the left half of the table evaluate the nowcasting quality (Objective O1), while the metrics in the right half evaluate the infection-day prediction quality (Objective O2). Latter metrics are not computed on the training subset as it requires the full hazard trajectory which is not available due to the use of windowed training.

The remarkably large difference between AUROC and AUPRC is due to the low prevalence of an infection event in the dataset. The AUROC is not sensitive to class imbalance and can still be high even if the model performs poorly on the positive class. In contrast, the AUPRC is more sensitive to class imbalance and provides a more informative measure of performance on the positive class, which is reflected in the much lower values compared to AUROC. At first sight, an $\text{AUPRC} \approx 0.36$ seems to be rather poor, but it is actually quite good given the low prevalence of infection events at about 0.7 % in the dataset. A random model would achieve an AUPRC of about 0.007, so the observed AUPRC values indicate a significant improvement over random guessing by a factor of at least 50.

All SpreadNet variants are able to predict the correct infection day of plants with an MAE of around 4 days. Since the infection day estimation is based on SugarViT’s prediction on roughly weekly intervals, this is quite a good performance. The IBS values are also low, indicating that the predicted hazard trajectories closely match the empirical survival patterns over time. The C-index values of at least 0.97 suggest that the models are very good at ranking plants according to their risk of infection, with those infected earlier generally receiving higher risk scores.

Further analyses are done with the SpreadNet M model, since it achieves the overall best performance. For inference, the model with the best validation BS is chosen. For the following analyses, the training subset is evaluated with all sampling and augmentation parameters as specified before. Solely, the random window choice is replaced by full-series sampling so that it matches the validation and testing methods.

In Section 4.7.4, it was mentioned that SpreadNet is trained on random data window lengths $\tilde{D} \sim \mathcal{U}(3, 10)$. Inside this length range, one can observe a window length dependency of the event-day MAE as shown in Figure 4.8. Particularly, the best test data performance is achieved for a window length of 5 days. This is why further analysis is done with this window length. The model is trained on random window lengths, which allows it to learn to make predictions based on varying amounts of historical data. However, there may be an optimal window length that provides the best balance between having enough historical information for accurate predictions and avoiding overfitting to longer sequences. The observed performance peak at a window length of 5 days suggests that this length may provide the most relevant information for predicting infection timing while still allowing the model to generalize well to unseen data.

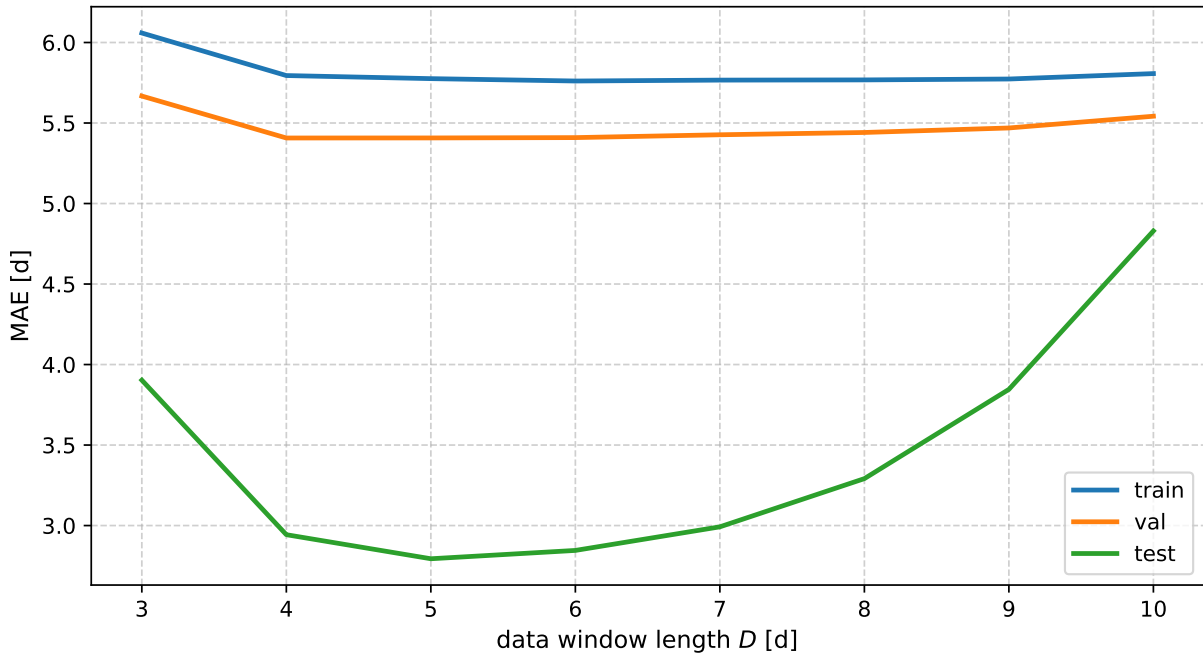


Figure 4.8: MAE for event day vs. data window length. The trained SpreadNet M model is evaluated on the MAE for predicting the event day as a function of the data window length D used for prediction. The lines represent the training, validation, and test subsets.

Besides the raw quantization of the model’s performance by defined metrics, it is desirable to understand the model’s behavior and whether it captures meaningful patterns in the data. This is why the focus of the next section is on the analysis of input saliencies and attention maps.

4.8.1.2 Saliency and Attention Maps

Complementary to the previously discussed attention maps, saliency maps are commonly used to analyze the relevance of input features by computing the gradient of the model's output with respect to its input. In order to obtain a more stable and informative saliency measure, the mean absolute value of the product of the gradient and the input is computed, which is often referred to as "gradx". This method helps to identify which input features have the most influence on the model's predictions. Furthermore, the saliency is accumulated by calculating the mean over all samples. Hence, the (gradx) saliency is defined as

$$\text{Sal}_{\text{gradx}}^{(j)} := \frac{1}{N} \sum_{i=1}^N \left| x_{i,j} \frac{\partial f(\mathbf{x}_i)}{\partial x_{i,j}} \right|, \quad (4.67)$$

where N is the number of samples, $\mathbf{x}_i$ is the i -th input, j the feature index, and f is the SpreadNet model. In vectorized form and element-wise multiplication ($\odot$), this can be written as

$$\text{Sal}_{\text{gradx}} := \frac{1}{N} \sum_{i=1}^N |\mathbf{x}_i \odot \nabla_{\mathbf{x}_i} f(\mathbf{x}_i)|, \quad (4.68)$$

resulting in so-called *saliency maps*.

Aggregated saliency maps provide insights into which features are most influential for the model's predictions across the entire dataset. Figure 4.9 shows the input saliencies for all features aggregated over meaningful feature groups or windows.

For the environmental features at the ego plant, three meaningful aggregations are possible. (1) The mean saliency over all hours and days shows a focus on environmental features of the night (0-2), early morning (5-6), noon (11-13), and evening (19-22) hours. The most relevant day is the current day and the relevance of past days is continuously decreasing. In the training and validation subset, a focus on the night hours of the days -2 and -3 can be observed, which is not visible in the test subset. (2) When the daily hours are accumulated into day- and feature-wise saliencies, the most relevant features are wind speed and spore concentration. This makes perfect sense with regard to the expected disease spread dynamics of fungal diseases, where infectious spores are mainly dispersed by the wind. (3) An accumulation of the days into hourly and feature-wise saliencies, moreover, shows wind speed and spore concentration during the mid-night and morning hours are most relevant. Additionally, the spore concentration in the evening hours is relevant.

For the ego-plant features, patterns look quite different comparing the training/validation and test subset. As in (2), the accumulation is done by features and days. In the test subset, the most important features are the current-day values for the BBCH stage,

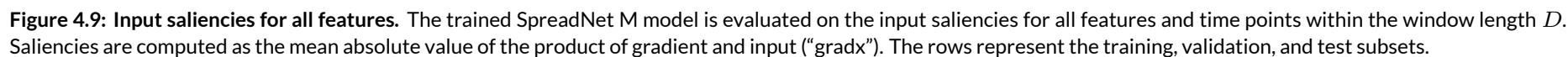
cover ratio, plant radius and the DIV. Cover ratio and BBCH are directly inferred from the GDD but seem to be more relevant than the GDD itself. Mainly, the current day and the last two days are important. For the training and validation subset, however, a focus is on the ETo with increasing relevance for past days. A phytopathological explanation for this could be that the plant’s water stress level, which is captured by the ETo, is an important factor for the infection risk. The relevance of past days’ ETo values could indicate that the plant’s water stress history has an influence on its susceptibility to infection. But since this pattern is not visible in the test subset, it might be a spurious correlation that does not generalize well to unseen data. Among all ego-plant features, the CLS fungicide treatment feature seems to be the least relevant, which could indicate that the model does not find a strong relationship between the fungicide treatment and the infection risk in the given dataset. This could be due to various reasons, such as the fungicide being applied too late to have an effect, or the disease pressure being so high that the fungicide’s protective effect is not sufficient to significantly reduce the infection risk. However, further analysis would be needed to confirm this hypothesis and to understand the underlying reasons for this pattern.

Performing the same accumulation for the neighbor features, the most relevant features are the cover ratio, the incubation and fructification cycles, and the plant radius. This makes sense since the cover ratio and plant radius are indicators of the plant’s size and density, which can influence the likelihood of spore deposition and infection by contact of leaves. The incubation and fructification cycles are direct indicators of the disease progression in the neighbor plants and, thus, have strong influence on the infection risk for the ego plant. The relative location of the neighbor plant seems to be less relevant, which could indicate that the model is not interested in the spatial pattern but only on the raw distance of neighbors which is already encoded in the neighbor index.

Now, the attention-based model parts of SpreadNet are analyzed. Firstly, the attention weights for the cross-day self-attention pooling (cf. Section 4.7.2.2) are shown in Figure 4.10. The attention weights describe how much each hour contributes to the pooling of hourly data into daily data. This allows the weights to serve as an indication of how important or unimportant the model considers individual hours or periods of the day to be. At least for the data fusion of the environmental parameters with the daily parameters, the model seems to emphasize evening and night hours. This could be a hint to the fact that the disease spread dynamics are more active during these hours, or that the environmental conditions during these hours are more indicative of infection risk. However, further analysis is needed to confirm this hypothesis and to understand the underlying reasons for this pattern. In any case, the model estimates that these hours are particularly representative for the environmental data.

Another interesting observation can be done by analyzing the attention weights for the ego-conditioned neighbor attention module (cf. Section 4.7.2.4). These attention weights describe how much each neighbor contributes to the prediction for the ego plant. This allows the weights to serve as an indication of how important or unimportant the model considers individual neighbors to be. Since the neighbors are sorted by distance to the ego plant, the weights can give a hint on the influence radius of the disease. The plot in Figure 4.11 shows that the model assigns higher attention weights to neighbors that are closer to the ego plant, which is consistent with the expectation that closer neighbors have a stronger influence on the infection risk of the ego plant. After about 4 neighbors, the attention weight is at a low, nearly constant value indicating that those 4 neighbors are considered to be the main drivers of the infection risk. Furthermore, this gives a hint on the influence radius of the disease, which can be made visible by calculating the distance of the neighbors. The lower plot in Figure 4.11 shows that the 4th neighbor (index 3) is on average about 45 cm away from the ego plant.

In order to check if the model is capturing the overall infestation stage of the fields, the cumulative number of infected plants over time can be analyzed. Equation (4.64) is used to compute the predicted infection day and the cumulative number of infected plants for each day after sowing is collected. This allows to compare the true and predicted cumulative infected plants over the harvest cycle. Figure 4.12 shows that the model captures the overall progression of the disease infestation well, as the predicted cumulative sums closely follow the true cumulative sum of infected plants. For the test dataset, the model seems to predict the first infections to happen earlier than they actually happen in the true data. However, the overall shape of the curve is still quite similar, indicating that the model captures the general dynamics of the disease spread, even if there are some discrepancies in the timing of the initial infections. Additionally, the test dataset is rather small, so the observed pattern might be not very representative for the overall generalization capabilities of SpreadNet. Furthermore, one has to state that this analysis is based on the true data and not on the simulation, where the model’s predictions directly influence the future data. This would lead to new and possibly “difficult” data points for the model, which can cause a butterfly effect and thus to diverging outcomes in the simulation over time. This is why the next section focuses on the evaluation of SpreadNet’s performance in an autoregressive simulation setup.



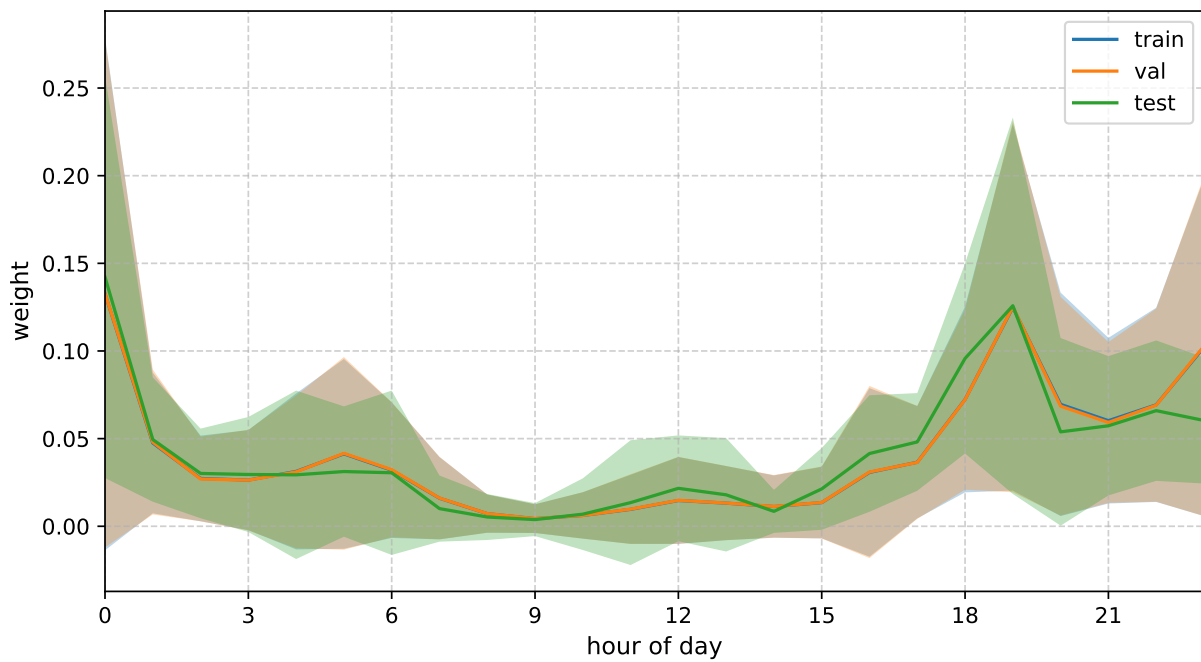


Figure 4.10: Attention weights for the cross-day self-attention pooling module. The plot shows the hourly pooling weights for the SpreadNet module described in Section 4.7.2.2 for all subsets. The mean and std denoted by the shaded area is computed by averaging over all datasets. Differences between the training and validation subset are barely visible, while the test subset shows a slightly different pattern.

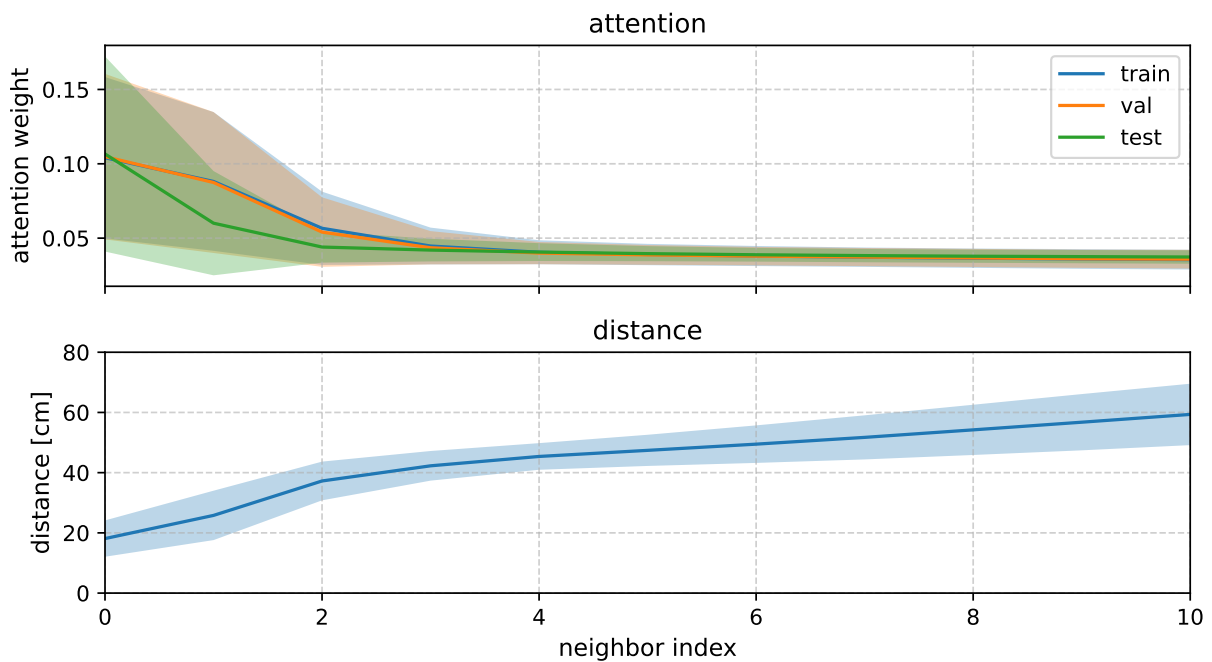


Figure 4.11: Attention weights for ego-conditioned neighbor attention module and mean neighbor distance. The upper plot shows the attention weights for the SpreadNet module described in Section 4.7.2.4. Along with it, the lower plot shows the distance of the neighbor index to the ego plant. The mean and std denoted by the line and the shaded area, respectively, is computed by averaging over all datasets.

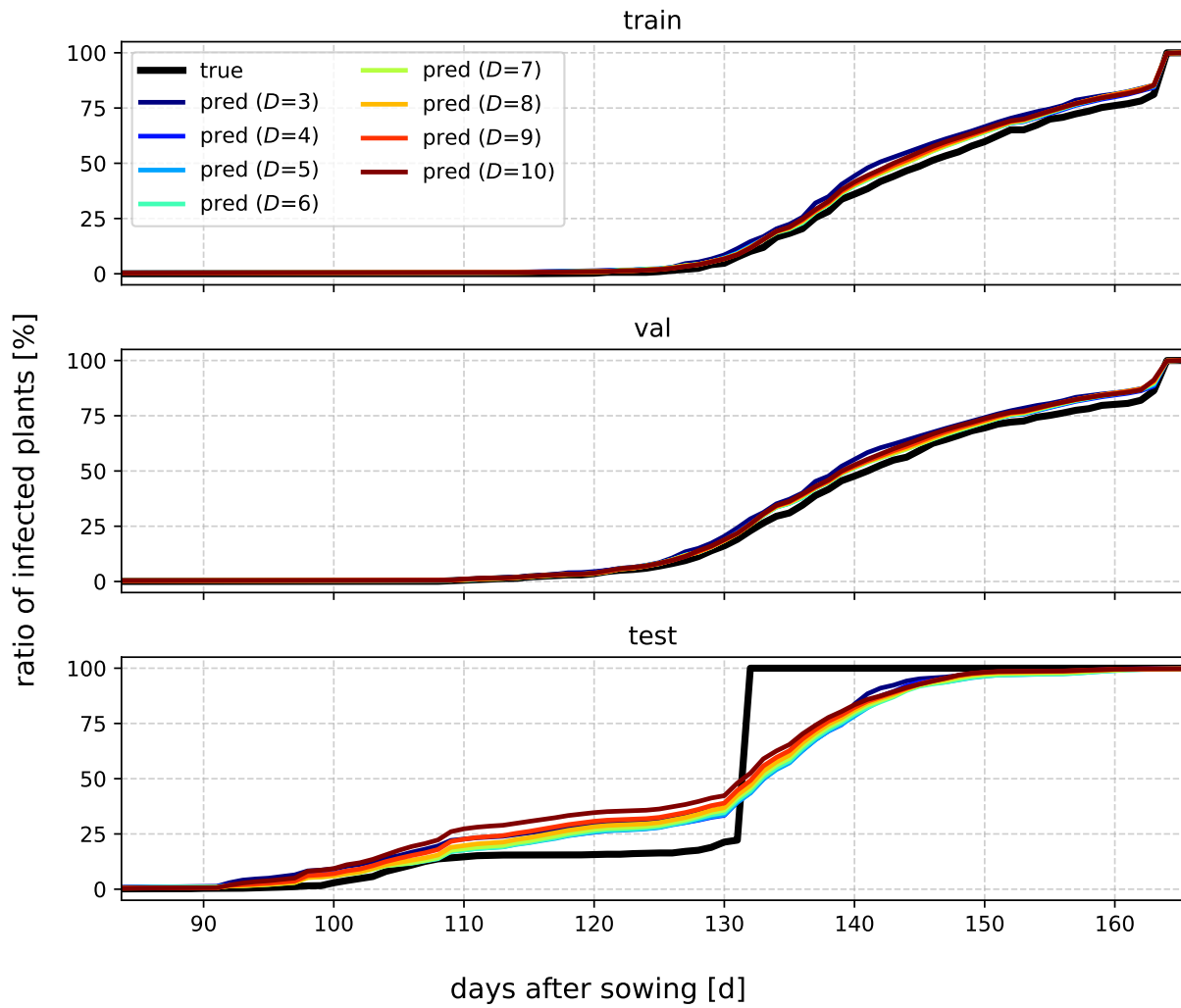


Figure 4.12: Cumulative infected plants over the harvest cycle. The true (thick line) and predicted (thin lines) cumulative infected plants show the progression of the disease infestation over time. The different predictions are based on different data window lengths D that SpreadNet uses for the prediction.

4.8.2 Evaluation on the Simulation

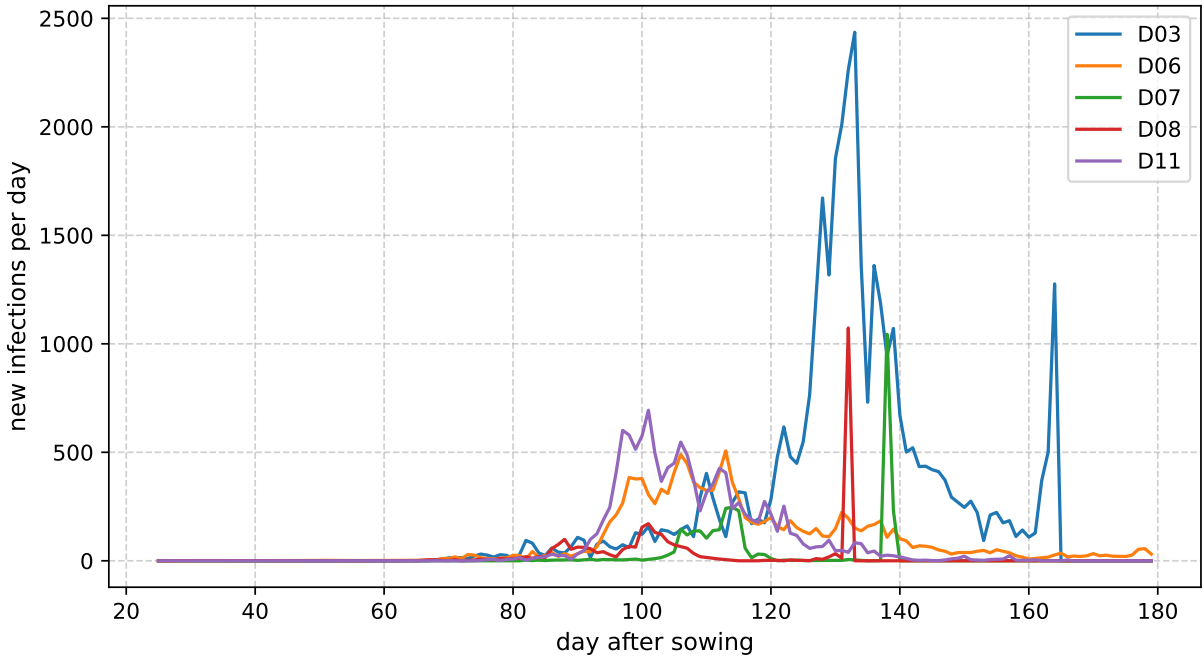


Figure 4.13: New infections per day for all datasets. The plot shows the (true) number of new infections per day after sowing for all datasets used for the spread modeling. Early and late days without new infections are not shown. The different colors represent the different datasets.

The model has been trained on “fixed” data, where the ego plants are basically independent of each other since wrong predictions on the ego plant do not affect the plants parameters as a neighbor for other ego plants. However, in an agent-based simulation setup, where SpreadNet is used as an autoregressive component, the prediction for an ego plant indeed has direct influence on plant agents in its neighborhood. In this section, the performance of SpreadNet is, therefore, evaluated on how stable the prediction quality is over time when applied side-by-side to the agent-based simulation. Thereby, one is able to quantify what is further called *nowcasting decay*. For this purpose, a target-day autonomy analysis is performed. Let d_t be the target day on which the prediction quality of SpreadNet should be analyzed. An *autonomy window* $\mathcal{A} = \{d_t - a, \dots, d_t\}$ with length $a + 1$ and $a \geq 0$ is defined on which the simulation is rolled out by using SpreadNet as infection model. The days $\{1, \dots, d_t - a - 1\}$ are simulated by propagating the plant development and using the environmental data and the infection days from the original dataset. Thus, the larger a , the more days are predicted by SpreadNet autonomously. In the following analysis, experiments with $a \in [0, 5]$ are performed. Furthermore, 4 independent rollouts are conducted in order to reduce the influence of stochasticity on results. In order to have a more solid data basis for the simulation, the analysis is performed on the dataset D06. It was used for training and validation of SpreadNet, but

it contains more plants and more variability in terms of treatment schemes and spread dynamics than the test dataset D08 (cf. Table 1.3).

Figure 4.13 shows the number of new infections per day after sowing for all datasets used for the spread modeling. For D06—the yellow curve in the figure—the most dynamic time period regarding infection events is roughly between day 80 and day 160. Thus, the target days d_t are chosen to be in this range with a stride length of 2 days, resulting in 41 investigated target days.

Since the infection simulation is basically a binary classification, one can count *true positive* (TP), *false positive* (FP), *false negative* (FN), and *true negative* (TN) infection predictions to compute further classification quality metrics. Let $\mathcal{J}(d_t) = \{i \mid d_{\text{inf}}^{(i)} = d_t\}$ be the set of (true) infected plants at day d_t and $\hat{\mathcal{J}}(d_t, a, r) = \{i \mid \hat{d}_{\text{inf}}^{(i)}(a, r) = d_t\}$ be the set of predicted infected plants at day d_t for the autonomy window a and rollout index r . Then, the confusion matrix components are defined as

$$\begin{aligned} TP(d_t, a, r) &:= |\mathcal{J}(d_t) \cap \hat{\mathcal{J}}(d_t, a, r)|, & FP(d_t, a, r) &:= |\hat{\mathcal{J}}(d_t, a, r) \setminus \mathcal{J}(d_t)|, \\ FN(d_t, a, r) &:= |\mathcal{J}(d_t) \setminus \hat{\mathcal{J}}(d_t, a, r)|, & TN(d_t, a, r) &:= |\overline{\mathcal{J}}(d_t) \cap \overline{\hat{\mathcal{J}}}(d_t, a, r)|, \end{aligned} \quad (4.69)$$

where the overlined sets denote the complement of the respective sets, i.e., the healthy plants. For the quality evaluation, the following metrics are used:

True rates For all target days d_t , autonomy windows a , and rollouts r , the true positive rate (TPR) and true negative rate (TNR) are computed as

$$\begin{aligned} \text{TPR}(d_t, a, r) &:= \frac{TP(d_t, a, r)}{TP(d_t, a, r) + FN(d_t, a, r)}, \\ \text{TNR}(d_t, a, r) &:= \frac{TN(d_t, a, r)}{TN(d_t, a, r) + FP(d_t, a, r)}. \end{aligned} \quad (4.70)$$

The TPR (also known as sensitivity or recall) measures the proportion of actual infected plants that are correctly identified by the model, while the TNR (also known as specificity) measures the proportion of actual healthy plants that are correctly identified. With other words, the TPR indicates how well the model can identify infected plants, while the TNR indicates how well the model can identify healthy plants.

Error rates For all target days d_t , autonomy windows a , and rollouts r , the false positive rate (FPR) and false negative rate (FNR) are computed as

$$\begin{aligned} \text{FPR}(d_t, a, r) &:= \frac{FP(d_t, a, r)}{FP(d_t, a, r) + TN(d_t, a, r)}, \\ \text{FNR}(d_t, a, r) &:= \frac{FN(d_t, a, r)}{TP(d_t, a, r) + FN(d_t, a, r)}. \end{aligned} \quad (4.71)$$

The FPR measures the proportion of healthy plants that are incorrectly identified as infected by the model, while the FNR measures the proportion of infected plants that are incorrectly identified as healthy. With other words, the FPR indicates how often the model falsely identifies healthy plants as infected, while the FNR indicates how often the model falsely identifies infected plants as healthy.

Diagnostic odds ratio In medical testing, the diagnostic odds ratio (DOR) is a commonly used measure of the effectiveness of a diagnostic test. From the perspective of this use case, it is defined as the ratio of the odds that an infected plant is correctly identified as infected to the odds that a healthy plant is incorrectly identified as infected. Thus, it measures how well the model can discriminate between infected and healthy plants. The DOR is computed as

$$\text{DOR}(d_t, a, r) := \frac{\frac{TP(d_t, a, r)}{FN(d_t, a, r)}}{\frac{FP(d_t, a, r)}{TN(d_t, a, r)}} = \frac{TP(d_t, a, r) \cdot TN(d_t, a, r)}{FP(d_t, a, r) \cdot FN(d_t, a, r)}. \quad (4.72)$$

If $\text{DOR} = 1$, the model is not better than random guessing. For $\text{DOR} > 1$, the model's discrimination capability increases with higher values.

Figure 4.14 shows the aggregation of each of the above metrics versus the autonomy window length a . Each metric $M \in \{\text{TPR}, \text{TNR}, \text{FPR}, \text{FNR}, \text{DOR}\}$ is shown by computing the mean and std over the rollouts $r \in \mathcal{R}$ and target days $d_t \in \mathcal{D}_t$. Thus, it is

$$\overline{M}(a) := \frac{1}{|\mathcal{R}||\mathcal{D}_t|} \sum_{r \in \mathcal{R}} \sum_{d_t \in \mathcal{D}_t} M(d_t, a, r), \quad (4.73)$$

and the corresponding sample stds are computed in the common way by

$$\sigma_M(a) := \sqrt{\frac{1}{|\mathcal{R}||\mathcal{D}_t| - 1} \sum_{r \in \mathcal{R}} \sum_{d_t \in \mathcal{D}_t} (M(d_t, a, r) - \overline{M}(a))^2}. \quad (4.74)$$

It can be observed that the overall TPR is rather low, while the FPR is rather high. This is not surprising given the low prevalence of infection events in the dataset, which makes it easier for the model to correctly identify healthy plants than infected plants. However, SpreadNet is still too conservative in its predictions. With increasing autonomy

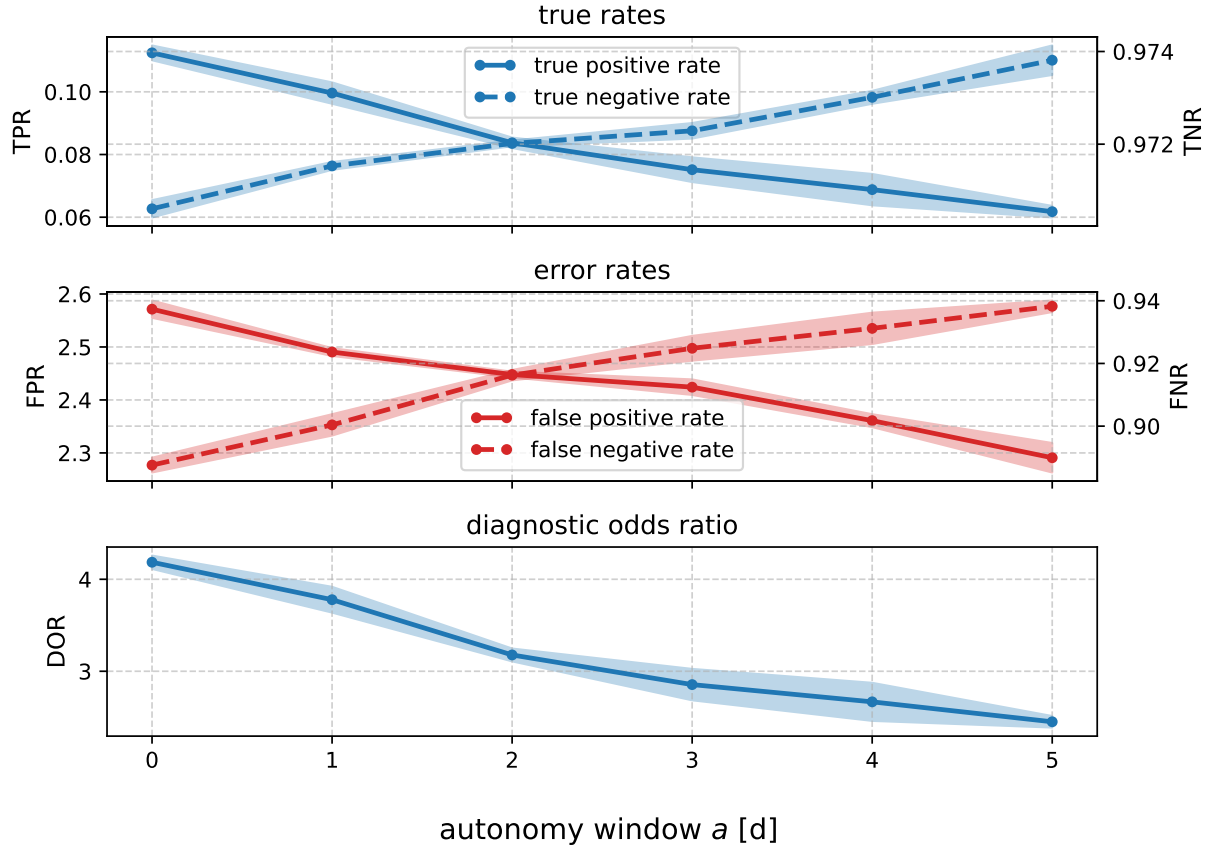


Figure 4.14: Target-day autonomy analysis summary. The plots show the simulation evaluation results by mean and std (shaded area) of the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), false negative rate (FNR), and diagnostic odds ratio (DOR) for different autonomy window lengths a . The mean and std are computed by averaging over all rollouts and target days.

window a , the TPR and FPR decrease, while the TNR and FNR increase. This behavior indicates that—as the autoregressive horizon grows—SpreadNet becomes even more conservative in predicting infections; it labels plants as non-infected more frequently. The decrease in TPR and the increase in FNR show that SpreadNet misses an increasing fraction of truly infected plants, causing loss of sensitivity. The decrease in FPR and increase in TNR show that SpreadNet raises fewer false alarms and correctly identifies non-infected individuals more often resulting in a specificity gain. The fact that—simultaneously—the DOR nevertheless decreases, implies that the loss in sensitivity outweighs the gain in specificity so that the overall ability to discriminate between infected and non-infected individuals deteriorates with increasing autonomy.

From a modelling perspective, this can be interpreted as a consequence of using the model outside the distribution it was trained on. During training, the model only sees true infection histories; during simulation with $a > 0$, it receives its own (possibly biased) past predictions as input. If the model already has a tendency to underpredict

infections, early missed infections lead to a simulated history with too few infected individuals. This in turn reduces the model's future predicted infection probabilities, reinforcing the negative bias over time. As a increases, this feedback loop becomes stronger, leading to systematically underestimated infection prevalence and degraded diagnostic performance in the autoregressive setting.

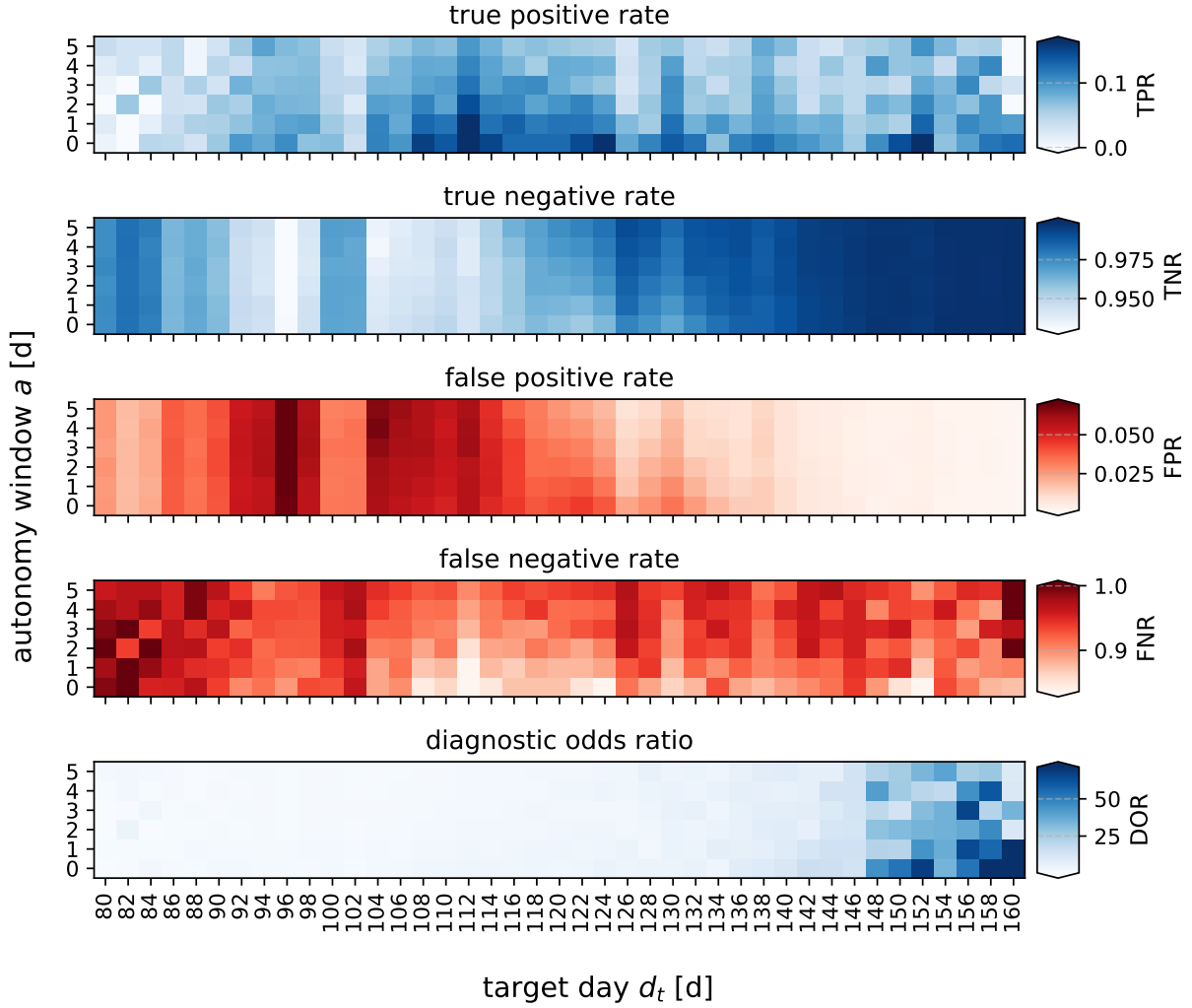


Figure 4.15: Target-day autonomy analysis heatmaps. The heatmaps show the simulation evaluation results by the mean true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), false negative rate (FNR), and diagnostic odds ratio (DOR) for different autonomy window lengths a and target days d_t . The mean is computed by averaging over all rollouts.

In order to get a more detailed picture of this increasing conservatism against autonomy window length, the same metrics as above are shown in Figure 4.15 but now only aggregated over the rollouts. Thus, it is

$$\overline{M}(d_t, a) := \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} M(d_t, a, r). \quad (4.75)$$

Since the infection dynamics and the model’s performance vary over time, it is insightful to analyze how the metrics evolve not only with the autonomy window length but also across different target days. This allows to identify if there are specific periods during the harvest cycle where the model’s performance degrades more rapidly with increasing autonomy.

The heatmaps reveal that this degradation is not homogeneous over time. For lower autonomy windows, the TPR is mostly higher, whereas the FNR tends to increase with larger autonomy windows over large parts of the analyzed period. This is consistent with the aggregated analysis above: once SpreadNet has to rely on its own predictions over several consecutive days, missed infections in the beginning of the rollout reduce the number of infectious neighbors in the simulated history. Consequently, the infection pressure in later simulated days is underestimated, which makes further infections less likely and primarily harms the detection of truly infected plants.

At the same time, the heatmaps show that this conservative bias can also have a positive effect: it can improve the classification of healthy plants. This is particularly visible for late target days, where the TNR increases and the FPR decreases. The effect is strongest for target days 148 – 160, where the DOR is clearly higher than in earlier phases. A plausible explanation is that, by this time, the main infection wave has already passed and the number of new infections is decreasing. In that situation, predicting fewer new infections is not only a model bias anymore, but also closer to the true disease dynamics. As a result, the model produces fewer false alarms, which improves TNR and reduces FPR. Even though the TPR still decreases with increasing autonomy window, the gain in specificity appears to outweigh this loss more strongly in the late season, which is reflected by the improved DOR values.

Overall, the heatmaps suggest that the autoregressive weakness of SpreadNet depends on the epidemic phase. During periods of active disease spread, larger autonomy windows mainly strengthen underprediction and therefore reduce sensitivity. Near the end of the season, however, the epidemic becomes less dynamic, and the same conservative behavior can even help by avoiding spurious late infections. This explains why small autonomy windows are generally better for preserving TPR, while larger autonomy windows can still improve TNR and FPR for some later target days.

4.9 Discussion

The analysis of prediction results has shown that SpreadNet is able to predict infection events and, thus, is able to detect favorable environmental and spatio-temporal conditions for disease spread dynamics. Moreover, the generalization capabilities look

promising even on the rather limited test dataset. Although SpreadNet uses some features that are derived from phytopathological knowledge, the learning process itself is completely data-driven. Nevertheless, the analysis of attention and saliency maps has revealed that the model captures meaningful patterns in the data that are at least explainable on a phytopathological level. This is a promising result, since it indicates that the model is not just learning spurious correlations but is actually “intelligent” in a way that its architecture allows to capture physical and biological laws.

However, the evaluation of SpreadNet in an autoregressive simulation setup has shown that the model’s performance degrades over time when it is used to generate its own input data. This is a common issue in models that are trained on fixed data and then applied in a sequential manner, as errors can accumulate and lead to a divergence from the true data distribution. This phenomenon, referred to as the butterfly effect, highlights the importance of training models in a way that they can generalize well to data generated by themselves, which is a challenging task. Future work could focus on improving the robustness of SpreadNet in such autoregressive settings by training it side-by-side with the agent-based simulation. This could involve techniques such as reinforcement learning or online learning, where the model is continuously updated based on the data it generates during the simulation.

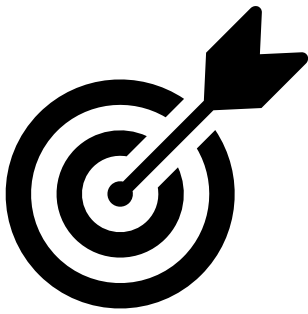
The datasets used in this work are rather controlled field experiments which hardly represent real-world conditions. Thus, more real-world data is probably needed to unleash the full potential of SpreadNet.

4.10 Interim Conclusion

Despite all possible limitations and open questions, it has already become apparent that SpreadNet training reveals interesting details about infection mechanisms shown by the saliency and attention weight analyses. It was shown that the large-scale DS annotations provided by SugarViT can be used to get insights into the disease spread dynamics if they are combined with environmental sensor data and expert knowledge provided by phytopathological features and physical modeling.

Moreover, the agent-based (or plant-based) modeling approach has a major advantage over classical epidemiological modeling approaches like S(E)I(R) models [5, 55, 72], since it allows capturing the heterogeneity of plants and their interactions in a more detailed way. This is particularly important for plant diseases, where the spatial structure of the field and the local interactions between plants can have a significant influence on the disease spread dynamics. The plant-based modeling approach allows capturing these local interactions and heterogeneities, which can lead to more accurate predictions and a better understanding of the underlying processes.

It can be stated that the modeling and training approach of SpreadNet leaves open questions for further research and improvements but already is a solid foundation and shows promising results. With availability of data from realistic field conditions in commercial agriculture, the potential of SpreadNet can be fully realized, enabling more accurate and actionable insights for disease management and crop protection. With its ability to capture complex spatio-temporal patterns and interactions between environment and plant/disease developments, SpreadNet could be used as a powerful tool for forecasting disease outbreaks, optimizing intervention strategies, and ultimately improving crop yields and sustainability in future agriculture.



5

Conclusion

In this final chapter, the main findings of all research steps are summarized and framed in the context of the overall research question. The limitations of the presented work are discussed and potential future research directions are outlined. Particular attention is given to future developments from both a technical-scientific and a socio-economic perspective.

5.1 Key Contributions

During all three main research steps, substantial insights into the combination of data-driven machine learning (ML) models and expert knowledge from the domain of plant pathology have been gained. Thus, the following section summarizes the key contributions of this dissertation regarding each of the three main research steps.

5.1.1 Plant Cataloging

For this initial step of the overall research work, it was shown that a simple approach based on a few parameters that can be easily measured in the field can already achieve good performance for plant cataloging. No trained ML model is needed for the extraction of plant positions. This is especially important for practical applications, where complex models may not be feasible due to limited computational resources or lack of

large annotated datasets. The proposed workflow is completely agnostic of the specific crop type, planting patterns, and image resolutions, which makes it flexible and easily adaptable to various scenarios. Moreover, it is semi-automated, meaning that it requires a small number of user-defined parameters. However, these parameters mostly have a physical scale so that they are easy to estimate and interpret for an application. This also ensures that the workflow can be easily adapted to different conditions and knowledge transfer is facilitated. Due to the modular and “black-box-free” nature of the workflow, the parameters used in a certain step can be easily adjusted and optimized for the specific application. Thus, the workflow becomes transparent and allows for a better understanding of the underlying processes, which makes it a valuable tool for plant cataloging in precision agriculture.

5.1.2 Disease Severity Prediction

This research step has shown that the combination of “cheap” data (unmanned aerial vehicle (UAV) imagery and environmental sensor data) and “expensive” data (human-expert disease severity (DS) ratings) can be used to conceptualize and train efficient models like SugarViT. In many common agricultural use cases, the acquisition of this expensive data is a major bottleneck for the development of data-driven models. Thus, cheap (meta) data from easy-to-use sources or sensors can be a valuable resource to overcome this bottleneck and enable the development of models that can be trained on large unlabeled or sparsely labeled datasets. The proposed model architecture is flexible and can be easily adapted to various tasks and data types, which makes it a powerful tool for precision agriculture applications. By using the attention mechanisms, the model focuses on the most relevant features in the data and outperforms convolutional architectures with comparable parametric complexity. Particularly for image processing, the vision transformer (ViT) architecture can capture long-range dependencies and global context which is shown to be beneficial for DS prediction. The integration of environmental sensor data further enhances the model’s ability to capture the complex relationships between the plant’s health and its surrounding environment, leading to improved prediction performance.

All in all, the DS prediction demonstrates how prior knowledge about the data and the problem can be effectively integrated into the model architecture and training process. Simultaneously, the model architecture is not limited to the specific task of DS prediction but can be easily adapted to various other tasks in precision agriculture and allows the model to learn which features are most relevant and how they are correlated, which are key aspects for the generalization capabilities of the model.

5.1.3 Disease Spread Modeling

The elaboration of SpreadNet has shown that it is possible to model the spatio-temporal dynamics of disease spread in plants by leveraging the ego-perspective of the plants and incorporating information from neighboring plants. The use of transformer architectures allows for capturing complex relationships among the data. Moreover, the model does not require an underlying constellation of the plants—e.g., a graph—to be assumed, but rather allows for flexible integration of neighboring plant information, which increases the applicability of the model to various scenarios. By this approach, the model is local and can be easily applied to large fields without the need for a global model that captures the entire field. This is especially important for practical applications where computational resources may be limited and/or the data is sparse. The proposed model architecture is flexible and can be easily adapted to various tasks and features. Using multiple layers of attention mechanisms with different physical purposes, the model has a meaningful inductive bias for the problem of disease spread modeling. This not only facilitates the integration of expert knowledge and physical understanding of the problem, but also provides interpretable insights into the model’s decision-making process. Furthermore, the loss function is designed to respect not only the accuracy of the nowcasting predictions but also the conditional survival probabilities. This dual focus is necessary to deal with the challenge of the low event prevalence and the resulting imbalances to the infection event data.

All in all, the spread modeling demonstrates how prior knowledge about the data and the problem can be effectively integrated into the model architecture and the loss function. Simultaneously, the model architecture is not limited to the disease spread modeling of *Cercospora* leaf spot (CLS) but can be easily adapted to various other settings with different feature sets. The model is allowed to freely learn, prioritize, and combine connections and correlations; however, it is always bound by the physical constraints and biological conditions of the system. These aspects are crucial for the generalization capabilities of the model and—as a result—the phytopathological knowledge transfer among various diseases on various crops. As an outcome of this research step, an intelligent system is gained that can “understand” the spread dynamics and can assist in the decision-making process for disease management. Furthermore, it educates the user about the underlying spread dynamics and thus fosters the understanding of the disease spread process, which is an important aspect for the adoption of such a system in practice.

5.1.4 End-to-End Workflow

If all the above steps are combined, a powerful end-to-end workflow is gained that can be applied to various disease spread studies. Starting with the plant cataloging step, no prior data but UAV orthophotos are needed to build a solid data basis for later knowledge-guided ML steps. The workflow is designed to be intuitive from the ground up, thanks to intuitive, flexible, and interpretable model inputs and outputs, which is crucial for the practical application—also for non-experts in ML and data science. Due to its modular nature, the workflow can be easily adapted to various scenarios and use cases. Furthermore, it is not limited to the specific crop type, disease, or data types used in this dissertation but can be easily adapted to various other settings, as long as the plants are spatially separable in at least some orthophotos and some expert labels are available for the DS prediction step.

In this sense, the contribution of the workflow is not limited to the sequential combination of individual modeling steps. It rather establishes a coherent framework in which data acquisition, plant-level structuring, disease assessment, and spread-aware prediction are linked in a way that remains transparent and practically interpretable throughout the process. This is particularly relevant for agricultural applications, where model performance alone is not sufficient, but traceability, modular adaptability, and the possibility of deployment under constrained field conditions are equally important prerequisites for adoption [91, 166].

5.2 Limitations

Despite the promising results throughout all steps, some limitations of the presented workflow were identified and need to be discussed. These limitations can be categorized into methodological and data-related limitations.

5.2.1 Methodology

Most of the workflow steps in the plant cataloging in Chapter 2 are based on simple computer vision (CV)-based image processing techniques. While this has the advantage of being simple, robust, and interpretable, it also has some limitations. For instance, the workflow may not work well for very small plants, heavily overlapping plants, or certain sowing patterns. Moreover, the workflow is based on a few parameters that need to be estimated by the user, which may introduce some subjectivity and variability in the results. Although most of the steps are agnostic to the specific sowing pattern, steps like the seeding line detection (cf. Section 2.4.3.2), weed filtering (cf. Section 2.4.3.3), or plant re-indexing (cf. Section 2.4.3.5) assume a conventional sowing with more or less straight

seeding lines. This is by far the most common sowing pattern in productive agriculture, so in most cases, this assumption is not a problem. However, for certain crops and special sowing patterns like intercropping or mixed cropping, some steps may not work well. Nevertheless, due to the modular style of the workflow, the other steps still work, and problematic steps can be easily adapted or excluded to fit the specific sowing pattern. Alternatively, the field can be split into smaller subfields with more conventional sowing patterns, and the workflow can be applied to each subfield separately.

The DS prediction with its SugarViT model is mainly driven by UAV data. This has the advantage of being a cheap and easy-to-use data source, but it also has some limitations. For instance, the DS prediction may not work well for very early stages of the disease when the symptoms are not yet visible from the air. Moreover, the DS prediction is based on a few expert labels that need to be estimated by the user, which may introduce some subjectivity and variability in the results. The label distribution learning (LDL) approach already mitigates subjectivity by introducing label uncertainty, but it cannot guarantee full subjectivity elimination. In addition, the uncertainty modeling used for the expert labels still relies on a simplified assumption. The target distributions are derived from a fixed and symmetric uncertainty pattern, whereas real rating uncertainty may depend on the disease stage, symptom visibility, rater experience, and contextual factors in the field. As a result, uncertainty can be class-dependent, skewed, or even multimodal, which is not fully captured by the current formulation. This simplification is methodologically useful, but it may limit the fidelity with which human expert uncertainty is represented during training. Due to the fast evolution of the visual disease symptoms, data is always scarcer in the middle-range than in high and low DS levels. Thus, the model may have difficulties to learn to predict these levels accurately. One could mitigate this effect by interpolation or an exponential fit to the data, as performed in Section 4.3. So far, SugarViT is not a spatio-temporal model, which may limit its performance for DS prediction. The time information is only implicitly captured by the cheap data levels in the form of growing degree days (GDDs). Since the plant cataloging step provides full image time series, a temporal extension of SugarViT is possible and may further improve its performance. The attention map results show that the model focuses on the most relevant features in the data, but it is not fully clear how the model combines these features to make its predictions. Thus, the interpretability of the model could be further improved by analyzing the attention maps in more detail and by using other interpretability techniques.

A major limitation of the disease spread model SpreadNet is the weak performance when it is used within an iterative simulation. This is a common problem for data-driven spread models, since the model output is fed back as input for the next time step, which can lead to error accumulation over time. Moreover, the model may encounter unseen data distributions during the simulation, which can further degrade its performance.

Thus, the current version of SpreadNet is mainly suitable for nowcasting, short-term forecasting, and identification of environmental conditions conducive to disease infection or spread. Since the performance on the nowcasting task is good, it can be assumed that this shortcoming is not related to the model itself, but rather to the way it was trained. As present, the training process does not include a feedback loop where model predictions are fed back to the model. This would prime the model to its autoregressive application. However, a training side-by-side with the simulation is substantially slower.

5.2.2 Data Availability

More serious than the methodological limitations in this dissertation are the limitations on the data side. The data collected in the context of this dissertation is undoubtedly comprehensive, and the acquisition process was laborious. However, the data lacks crucial characteristics, making it difficult to draw firm conclusions about the generalizability of the approaches developed on its basis. Firstly, the data is based on multi-purpose field experiments with specific treatments and conditions. On the one hand, this guarantees a high data diversification and thus a good basis for the development of data-driven models. On the other hand, the data is not very representative for real-world scenarios, since it is a highly controlled setup. Although multiple fields and four growing seasons are involved in the dataset, they all come from the same region and, more or less, the same sensors or measurement devices. Generalization to real-world scenarios, other climate conditions, other sensors, or other crops, is—therefore—difficult to assess with this dataset. This made it all the more important to tune or adapt the developed approaches and models as little as possible to the dataset, which led to comparatively general and—at least in principle—generalizable results. Moreover, the available data only partially reflects the operational and organizational complexity of real agricultural practice. Important sources of variation such as heterogeneous management routines, farm-specific intervention strategies, economic constraints, documentation quality, and varying technical infrastructures are only weakly represented in controlled experimental settings. Consequently, the present data basis is well suited for methodological development, but only to a limited extent for assessing how robustly the proposed workflow can be integrated into routine field practice [129, 167].

The next logical step is a systematic validation of the proposed workflow on broader and less controlled data sources. To assess and improve generalizability, future studies should include heterogeneous datasets across regions, seasons, crop systems, disease pressures, and sensing technologies [91, 166]. Such comparative evaluations can reveal

domain-specific weaknesses, support targeted model refinements, and test transferability under realistic field conditions. At the same time, they create opportunities to identify additional use cases beyond the scenarios covered in this dissertation. Against this background, the following section outlines key directions for future work.

5.3 Future Work

Building on the limitations discussed above, future research should be approached along two complementary perspectives. The first is a technical and scientific perspective, which concerns the further methodological development of the presented workflow, its validation under more heterogeneous conditions, and its extension toward more comprehensive and practically deployable intelligent systems. The second is a socio-economic perspective, which addresses the conditions under which such systems can create value in agricultural practice, including usability, trust, accessibility, training, and the availability of representative real-world data. Both perspectives are tightly interconnected: scientific progress without practical adoption remains of limited impact, while adoption without technically robust and generalizable methods is unlikely to be sustainable. Against this background, the following subsections outline the most relevant directions for future work.

5.3.1 Technical and Scientific Perspective

From a technical and scientific perspective, one central direction for future work is the systematic extension of the proposed workflow toward broader generalization, richer data integration, and more robust predictive capabilities. As discussed in the previous section, the current approaches were developed on a comprehensive but still comparatively controlled data basis. A first priority is therefore the validation of all workflow components on more heterogeneous datasets covering different regions, seasons, crop systems, disease pressures, imaging configurations, and sensor technologies. Such evaluations are necessary not only to quantify transferability, but also to identify which parts of the workflow are truly general and which require adaptation to specific domains.

Beyond validation, future research should further investigate the role of foundation models (FMs) in agricultural phenotyping and plant health monitoring. Recent work indicates that agriculture can strongly benefit from large, transferable models that are pretrained across broad visual and multimodal domains and subsequently adapted to specialized downstream tasks [99, 162]. In the context of this dissertation, such models could provide a promising basis for DS estimation, plant organ detection, plant counting, and related perception tasks. In particular, a pretrained FM could reduce the dependence on large amounts of task-specific annotation and improve robustness under

varying field conditions, illumination patterns, cultivars, and sensor setups. For the DS task, this opens the possibility of combining the current knowledge-driven modeling strategy with large-scale pretrained representations in order to preserve interpretability and domain alignment while improving data efficiency and transferability.

A further important research direction concerns multimodal learning. The results of this dissertation already show that the combination of visual and environmental information is beneficial. However, future models should integrate these modalities more comprehensively and in a more explicitly structured way. This includes not only image data and environmental sensor measurements, but also plant-level descriptors, temporal context, management information, and possibly remote sensing (RS) observations from aerial or satellite platforms. Such *sensor fusion* could yield a more holistic representation of crop health and disease development, thereby improving both DS prediction and spread modeling. In particular, integrating temporally resolved image sequences with environmental trajectories may enable models to capture latent disease dynamics earlier and more reliably than with static snapshots alone.

In the case of disease spread modeling, future work should also move beyond the current focus on single-disease nowcasting toward more comprehensive and intervention-aware forecasting settings. This includes the joint modeling of multiple diseases, the explicit representation of competing or interacting stress factors, and the inclusion of agronomic interventions such as fungicide applications, irrigation regimes, canopy management, or cultivar-specific resistance. An initial representation of fungicide application is already included in SpreadNet. More of those extensions would allow the model not only to describe disease dynamics under observed conditions, but also to support counterfactual analyses and scenario-based simulations. In practice, this would be highly valuable for decision support, because it would enable the assessment of possible management strategies before they are implemented in the field.

Closely related to this is the need to improve the autoregressive simulation capabilities of SpreadNet. The current results suggest that the model is well suited for nowcasting and short-term forecasting, but less robust when repeatedly applied in iterative simulations. Future work should therefore address training strategies that expose the model to its own predictions during learning, for example through autoregressive training schemes, online adaptation, or simulation-coupled optimization. Although such procedures are computationally more demanding, they are essential if the long-term objective is to develop predictive systems that can simulate realistic spread scenarios over extended time horizons without severe error accumulation.

Another technically relevant direction is the translation of these methods into deployable systems for real-time or near-real-time agricultural decision support. Recent developments in cloud-edge-device collaboration and edge-enabled smart agriculture

show that modern agricultural intelligence systems increasingly rely on distributed computation across sensing devices, edge nodes, and cloud infrastructure [139, 163]. For the workflow developed in this dissertation, this implies that future work should not only optimize predictive performance, but also consider latency, computational efficiency, energy consumption, and robustness under limited connectivity. Lightweight model variants, staged inference pipelines, and hybrid cloud-edge architectures may therefore become crucial for transferring the proposed concepts from experimental research settings into regular agricultural operation.

Overall, the technical and scientific perspective points toward a future in which the presented workflow evolves from a set of well-founded task-specific models into a general, multimodal, scalable, and deployable framework for plant health monitoring and disease-aware crop management. Achieving this goal will require methodological advances, broader validation, and closer integration of agronomic knowledge, modern ML, and realistic deployment constraints.

5.3.2 Socio-Economic Perspective

From a socio-economic perspective, future work must address the question of how intelligent agricultural technologies can become accessible, trustworthy, and practically useful for the people who are expected to work with them. Precision agriculture does not fail primarily because of a lack of algorithms, but often because of barriers related to cost, complexity, infrastructure, training, interoperability, and uncertainty regarding actual benefits in day-to-day practice [129, 167]. Accordingly, the further development of data-driven systems for plant health monitoring should not be understood as a purely technical challenge, but as a broader transformation process that must account for social, organizational, and economic realities.

A first priority in this context is usability. Even highly accurate models will remain of limited value if their outputs are difficult to interpret or if their operation requires specialized technical expertise. Future work should therefore focus on interfaces and interaction concepts that present model outputs in a transparent, actionable, and agronomically meaningful way. This includes intuitive visualizations of uncertainty, disease progression, and risk levels, as well as recommendation formats that support rather than replace human decision-making. In particular, systems should be designed for users who may not have routine experience with artificial intelligence (AI) or ML, but who possess substantial experiential knowledge of crops, field conditions, and local management practices. The aim should not be to obscure agronomic reasoning behind automated predictions, but to create human-centered decision support systems that combine computational intelligence with the expertise of practitioners.

Closely connected to usability is the issue of trust. The adoption of digital technologies in agriculture depends strongly on whether users perceive them as reliable, understandable, and beneficial under practical conditions [129, 167]. For this reason, future work should investigate how explainability, uncertainty communication, and model transparency influence adoption behavior. In the context of this dissertation, the modular and knowledge-driven structure of the workflow already provides a favorable basis for such work, because it avoids purely opaque end-to-end automation. This characteristic should be preserved and expanded. Especially in sensitive decision contexts such as disease management, farmers and advisors need to understand not only what a system predicts, but also under which conditions these predictions are likely to be reliable and where their limits are.

Another major socio-economic challenge concerns education, training, and long-term support. The successful integration of data-driven systems into agricultural practice requires more than software delivery. It requires training concepts that enable users to interpret outputs correctly, assess uncertainties, and adapt workflows to their own operational settings. Future work should therefore include participatory development processes in which farmers, advisors, and researchers jointly evaluate prototypes, define relevant performance criteria, and identify barriers to adoption at an early stage. Such an approach would help to ensure that future systems are not only technically advanced, but also aligned with real informational needs, labor constraints, and farm-level decision processes.

The availability of representative real-world data is another issue that lies at the interface of scientific progress and socio-economic adoption. As already discussed, controlled experimental data is indispensable for method development, but it is not sufficient to establish practical trust and broad applicability. Future progress in data-driven plant disease modeling will therefore depend on creating larger and more diverse datasets from non-controlled field environments. This is not only a technical data acquisition task; it is also an organizational and economic one [129, 167]. Collecting such data requires stable collaborations between research institutions, farmers, technology providers, and possibly public agencies. Questions of standardization, incentives, data governance, privacy, ownership, and fair benefit sharing will become increasingly important as agricultural data infrastructures mature.

Finally, future work should also assess the broader economic and sustainability implications of intelligent plant health monitoring systems. Their value should not be measured exclusively by predictive accuracy, but also by their potential to improve resource efficiency, reduce unnecessary treatments, stabilize yields, lower labor intensity, and support environmentally responsible management decisions [19, 39]. In this sense, socio-economic evaluation should become an integral part of methodological research rather than an afterthought. Only if technological innovation is linked to measurable

agronomic, economic, and social benefit, it is possible to establish substantial adoption in precision agriculture.

In summary, the socio-economic perspective emphasizes that the future of data-driven plant disease management depends not only on better models, but also on better integration into agricultural practice. Accessibility, interpretability, training, collaboration, and data availability are therefore not peripheral concerns, but core prerequisites for the long-term impact of the scientific advances of knowledge-driven intelligent systems for precision agriculture.

List of Acronyms

AHE adaptive histogram equalization

AI artificial intelligence

AUPRC area under the precision-recall curve

AUROC area under the receiver operating characteristic curve

BBCH Biologische Bundesanstalt für Land- und Forstwirtschaft, Bundessortenamt und Chemische Industrie

BCE binary cross-entropy

BS Brier score

BSS Brier skill score

CLAHE contrast-limited adaptive histogram equalization

CLS Cercospora leaf spot

CNN convolutional neural network

CPD coherent point drift

CV computer vision

DAC days after canopy closure

DBSCAN density-based spatial clustering of applications with noise

DIV daily infection value

DLDL deep label distribution learning

DOR diagnostic odds ratio

DS disease severity

DSP daily spore production

ECE expected calibration error

ET_o reference evapotranspiration

FCL fully-connected layer

FFN feed-forward network
FM foundation model
FNR false negative rate
FPR false positive rate
GCP ground control point
GDD growing degree day
GELU Gaussian error linear unit
GIS geographic information system
GLI green leaf index
GLONASS global navigation satellite system
GMM Gaussian mixture model
GPS global positioning system
GPT generative pretrained transformer
GSD ground sampling distance
HE histogram equalization
IBS integrated Brier score
IfZ Institute of Sugar Beet Research
IoT internet of things
JDN Julian day number
KLD Kullback-Leibler divergence
KML keyhole markup language
LDL label distribution learning
LiDAR light detection and ranging
LN layer normalization
LWIR long-wave (thermal) infrared
MAE mean absolute error
MDO mean distribution overlap
ML machine learning
MLP multi-layer perceptron

MSE mean squared error
NDVI normalized difference vegetation index
NGRDI normalized green/red difference index
NIR near infrared
NLL negative log-likelihood
NLP natural language processing
NN neural network
NPG number of possible generations
OSAVI optimized soil-adjusted vegetation index
PDF probability density function
PMF probability mass function
R-CNN region-based convolutional neural network
REDGE red edge
ReLU rectified linear unit
RGB red, green, and blue channel
RMSE root mean squared error
RNN recurrent neural network
RS remote sensing
RTK real-time kinematic
SAR synthetic aperture radar
std standard deviation
SVM support vector machine
TNR true negative rate
TPR true positive rate
UAV unmanned aerial vehicle
UGV unmanned ground vehicle
UTM universal transverse Mercator
VI vegetation index
ViT vision transformer
WGS84 world geodetic system 1984

Own Publications

- [A] M. Günder, F. R. Ispizua Yamati, J. Kierdorf, R. Roscher, A.-K. Mahlein, and C. Bauckhage. “Agricultural Plant Cataloging and Establishment of a Data Framework from UAV-based Crop Images by Computer Vision”. In: *GigaScience* 11 (June 2022). giac054. doi: 10.1093/gigascience/giac054.
- [B] M. Günder, F. R. Ispizua Yamati, A.-K. Mahlein, and C. Bauckhage. “Towards Agent-based Disease Spread Modeling Combining Knowledge-driven Simulation and Machine Learning”. In: *2024 IEEE International Conference on Big Data (BigData)*. Dec. 2024, pp. 4692–4701. doi: 10.1109/BigData62323.2024.10825840.
- [C] M. Günder, N. Piatkowski, and C. Bauckhage. “Full Kullback-Leibler-divergence loss for hyperparameter-free Label Distribution Learning”. In: (Sept. 2022).
- [D] M. Günder, N. Piatkowski, L. Von Rueden, R. Sifa, and C. Bauckhage. “Towards Intelligent Food Waste Prevention: An Approach Using Scalable and Flexible Harvest Schedule Optimization With Evolutionary Algorithms”. In: *IEEE Access* 9 (2021), pp. 169044–169055. doi: 10.1109/ACCESS.2021.3137709.
- [E] M. Günder, F. R. I. Yamati, A. Barreto, A.-K. Mahlein, R. Sifa, and C. Bauckhage. “SugarViT—Multi-objective regression of UAV images with Vision Transformers and Deep Label Distribution Learning demonstrated on disease severity prediction in sugar beet”. In: *PLOS ONE* 20.2 (Feb. 2025), pp. 1–30. doi: 10.1371/journal.pone.0318097.
- [F] F. R. Ispizua Yamati, M. Günder, A. Barreto, et al. “Automatic Scoring of Rhizoctonia Crown and Root Rot Affected Sugar Beet Fields from Orthorectified UAV Images Using Machine Learning”. In: *Plant Disease* 108.3 (2024). PMID: 37755420, pp. 711–724. doi: 10.1094/PDIS-04-23-0779-RE.
- [G] F. R. Ispizua Yamati, M. Günder, C. Bauckhage, and A.-K. Mahlein. “Sensing the occurrence and dynamics of Cercospora leaf spot disease using UAV-supported image data and deep learning”. In: *Sugar Industry* 147.2 (2022), pp. 79–86. doi: 10.36961/si28345.
- [H] F. R. Ispizua Yamati, M. Günder, J. Bömer, et al. “Hybrid Modeling of Cercospora Leaf Spot Epidemiology: Integrating Mechanistic and Machine Learning Approaches Using Remote-Sensing and Environmental Data”. In: *Phytopathology®* (Dec. 2025). PMID: 41325634. doi: 10.1094/PHYTO-03-25-0113-R.
- [I] F. R. Ispizua Yamati, R. H. J. Heim, M. Günder, W. Gajda, and A.-K. Mahlein. “Image-to-image translation for satellite and UAV remote sensing: a use case for Cercospora Leaf Spot monitoring on sugar beet.” In: *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. 2023, pp. 783–787. doi: 10.1109/MetroAgriFor58484.2023.10424276.
- [J] J. Krause, M. Günder, D. Schulz, and R. Gruna. “New active learning algorithms for near-infrared spectroscopy in agricultural applications”. In: *at-Automatisierungstechnik* 69.4 (2021), pp. 297–306.

References

- [1] S. Abnar and W. Zuidema. “Quantifying Attention Flow in Transformers”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 4190–4197. doi: 10.18653/v1/2020.acl-main.385.
- [2] Agvolution GmbH. CLIMAVI One Data Sheet. 2023. URL: <https://www.climavi.eu/Datenblatt%3%A4tter/CLIMAVI%20Datenblatt%20-1.10.2023-.pdf> (visited on 11/14/2024).
- [3] R. G. Allen, L. S. Pereira, D. Raes, M. Smith, et al. “Crop evapotranspiration - Guidelines for computing crop water requirements - FAO Irrigation and drainage paper 56”. In: *FAO, Rome* 300.9 (1998), p. D05109.
- [4] Z. Al-Ameen, G. Sulong, A. Rehman, A. Al-Dhelaan, T. Saba, and M. Al-Rodhaan. “An innovative technique for contrast enhancement of computed tomography images using normalized gamma-corrected contrast-limited adaptive histogram equalization”. In: *EURASIP Journal on Advances in Signal Processing* 2015.1 (Apr. 2015). doi: 10.1186/s13634-015-0214-1.
- [5] R. M. Anderson and R. M. May. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press, May 1991. doi: 10.1093/oso/9780198545996.001.0001.
- [6] S. O. Araújo, R. S. Peres, J. C. Ramalho, F. Lidon, and J. Barata. “Machine Learning Applications in Agriculture: Current Trends, Challenges, and Future Perspectives”. In: *Agronomy* 13.12 (2023). doi: 10.3390/agronomy13122976.
- [7] M. F. Aslan, A. Durdu, K. Sabanci, E. Ropelewska, and S. S. Gültekin. “A Comprehensive Survey of the Recent Studies with UAV for Precision Agriculture in Open Fields and Greenhouses”. In: *Applied Sciences* 12.3 (Jan. 2022), p. 1047. doi: 10.3390/app12031047.
- [8] C. Atzberger. “Advances in Remote Sensing of Agriculture: Context Description, Existing Operational Monitoring Systems and Major Information Needs”. In: *Remote Sensing* 5.2 (2013), pp. 949–981. doi: 10.3390/rs5020949.
- [9] D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473 (2014).
- [10] A. Barreto, F. R. Ispizua Yamati, M. Varrelmann, S. Paulus, and A.-K. Mahlein. “Disease Incidence and Severity of Cercospora Leaf Spot in Sugar Beet Assessed by Multispectral Unmanned Aerial Images and Machine Learning”. In: *Plant Disease* 107.1 (2023). PMID: 35581914, pp. 188–200. doi: 10.1094/PDIS-12-21-2734-RE.
- [11] C. Bauckhage and K. Kersting. “Data Mining and Pattern Recognition in Agriculture”. In: *KI-Künstliche Intelligenz* 27.4 (2013), pp. 313–324.
- [12] H. Bleiholder and H. C. Weltzien. “Beiträge zur Epidemiologie von Cercospora beticola, Sacc. an Zuckerrübe - I. Die Inkubations- und die Fruktifikationszeit”. In: *Journal of Phytopathology* 72.4 (1971), pp. 344–353. doi: 10.1111/j.1439-0434.1971.tb03207.x.

- [13] H. Bleiholder and H. C. Weltzien. “Beiträge zur Epidemiologie von *Cercospora beticola* Sacc. an Zuckerrübe - II. Die Konidienbildung in Abhängigkeit von den Umweltbedingungen Temperatur, relative Luftfeuchtigkeit und Licht”. In: *Journal of Phytopathology* 73.1 (1972), pp. 46–68. doi: 10.1111/j.1439-0434.1972.tb02524.x.
- [14] H. Bleiholder and H. C. Weltzien. “Beiträge zur Epidemiologie von *Cercospora beticola* Sacc. an Zuckerrübe - III. Geopathologische Untersuchungen”. In: *Journal of Phytopathology* 73.2 (1972), pp. 93–114. doi: 10.1111/j.1439-0434.1972.tb02532.x.
- [15] C. H. Bock, G. H. Poole, P. E. Parker, and T. R. Gottwald. “Plant Disease Severity Estimated Visually, by Digital Photography and Image Analysis, and by Hyperspectral Imaging”. In: *Critical Reviews in Plant Sciences* 29.2 (2010), pp. 59–107. doi: 10.1080/07352681003617285.
- [16] C. H. Bock, J. G. A. Barbedo, A.-K. Mahlein, and E. M. D. Ponte. “A special issue on phytopathometry — visual assessment, remote sensing, and artificial intelligence in the twenty-first century”. In: *Tropical Plant Pathology* 47.1 (Feb. 2022), pp. 1–4. doi: 10.1007/s40858-022-00498-w.
- [17] C. H. Bock, J. G. A. Barbedo, E. M. D. Ponte, D. Bohnenkamp, and A.-K. Mahlein. “From visual estimates to fully automated sensor-based measurements of plant disease severity: status and challenges for improving accuracy”. In: *Phytopathology Research* 2.1 (Apr. 2020). doi: 10.1186/s42483-020-00049-8.
- [18] C. H. Bock, K.-S. Chiang, and E. M. D. Ponte. “Plant disease severity estimated visually: a century of research, best practices, and opportunities for improving methods and practices to maximize accuracy”. In: *Tropical Plant Pathology* (June 2021). doi: 10.1007/s40858-021-00439-z.
- [19] R. Bongiovanni and J. Lowenberg-Deboer. “Precision Agriculture and Sustainability”. In: *Precision Agriculture* 5.4 (Aug. 1, 2004), pp. 359–387. doi: 10.1023/B:PRAG.0000040806.39604.aa.
- [20] M. Bourriz, H. Hajji, A. Laamrani, et al. “Integration of Hyperspectral Imaging and AI Techniques for Crop Type Mapping: Present Status, Trends, and Challenges”. In: *Remote Sensing* 17.9 (2025). doi: 10.3390/rs17091574.
- [21] G. W. Brier. “Verification of Forecasts Expressed in Terms of Probability”. In: *Monthly Weather Review* 78.1 (1950), pp. 1–3. doi: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.
- [22] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer. “On Identifiability in Transformers”. In: *arXiv e-prints*, arXiv:1908.04211 (Aug. 2019), arXiv:1908.04211. doi: 10.48550/arXiv.1908.04211.
- [23] Y. Chen, E. Yan, J. Jiang, G. Zhang, and D. Mo. “An efficient approach to monitoring pine wilt disease severity based on random sampling plots and UAV imagery”. In: *Ecological Indicators* 156 (2023), p. 111215. doi: 10.1016/j.ecolind.2023.111215.
- [24] R. Chin, C. Catal, and A. Kassahun. “Plant disease detection using drones in precision agriculture”. In: *Precision Agriculture* 24.5 (Oct. 1, 2023), pp. 1663–1682. doi: 10.1007/s11119-023-10014-y.
- [25] W. Chivasa, O. Mutanga, and C. Biradar. “UAV-Based Multispectral Phenotyping for Disease Resistance to Accelerate Crop Improvement under Changing Climate Conditions”. In: *Remote Sensing* 12.15 (July 2020), p. 2445. doi: 10.3390/rs12152445.

- [26] A. Chlingaryan, S. Sukkarieh, and B. Whelan. “Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review”. In: *Computers and Electronics in Agriculture* 151 (2018), pp. 61–69. doi: 10.1016/j.compag.2018.05.012.
- [27] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. “What Does BERT Look at? An Analysis of BERT’s Attention”. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by T. Linzen, G. Chrupała, Y. Belinkov, and D. Hupkes. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 276–286. doi: 10.18653/v1/W19-4828.
- [28] H. Z. Cross and M. S. Zuber. “Prediction of Flowering Dates in Maize Based on Different Methods of Estimating Thermal Units”. In: *Agronomy Journal* 64.3 (1972), pp. 351–355. doi: 10.2134/agronj1972.00021962006400030029x.
- [29] J. Davis and M. Goadrich. “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240. doi: 10.1145/1143844.1143874.
- [30] A. Daw, R. Q. Thomas, C. C. Carey, J. S. Read, A. P. Appling, and A. Karpatne. “Physics-Guided Architecture (PGA) of Neural Networks for Quantifying Uncertainty in Lake Temperature Modeling”. In: *Proceedings of the 2020 SIAM International Conference on Data Mining (SDM)*. 2020, pp. 532–540. doi: 10.1137/1.9781611976236.60.
- [31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020. doi: 10.48550/ARXIV.2010.11929.
- [32] L. Drees, L. V. Junker-Frohn, J. Kierdorf, and R. Roscher. “Temporal prediction and evaluation of Brassica growth in the field using conditional generative adversarial networks”. In: *Computers and Electronics in Agriculture* 190 (2021), p. 106415. doi: 10.1016/j.compag.2021.106415.
- [33] R. O. Duda and P. E. Hart. “Use of the Hough Transformation to Detect Lines and Curves in Pictures”. In: *Commun. ACM* 15.1 (Jan. 1972), pp. 11–15. doi: 10.1145/361237.361242.
- [34] M. El Jarroudi, F. Chairi, L. Kouadio, K. Antoons, A. H. M. Sallah, and X. Fettweis. “Weather-based Predictive Modeling of Cercospora Beticola Infection Events in Sugar Beet in Belgium”. In: *Journal of Fungi* 7.9 (2021), pp. 11–13. doi: 10.3390/jof7090777.
- [35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 226–231.
- [36] Food and A. O. of the United Nations (FAO). The future of food and agriculture - Trends and challenges. en. Rome, Italy: FAO, Sept. 2017, p. 180.
- [37] Food and Agriculture Organization of the United Nations (FAO). FAOSTAT: Crop Production Statistics. 2026.
- [38] B. B. Gao, C. Xing, C. W. Xie, J. Wu, and X. Geng. “Deep Label Distribution Learning with Label Ambiguity”. In: *IEEE Transactions on Image Processing* 26.6 (2017), pp. 2825–2838. doi: 10.1109/TIP.2017.2689998.
- [39] R. Gebbers and V. I. Adamchuk. “Precision Agriculture and Food Security”. In: *Science* 327.5967 (2010), pp. 828–831. doi: 10.1126/science.1183899.
- [40] X. Geng. Label Distribution Learning. 2016.

- [41] X. Geng, X. Qian, Z. Huo, and Y. Zhang. “Head Pose Estimation Based on Multivariate Label Distribution”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 1974–1991. doi: 10.1109/TPAMI.2020.3029585.
- [42] X. Geng, C. Yin, and Z.-H. Zhou. “Facial Age Estimation by Learning from Label Distributions”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.10 (Oct. 2013), pp. 2401–2412. doi: 10.1109/tpami.2013.51.
- [43] H. C. J. Godfray, J. R. Beddington, I. R. Crute, et al. “Food Security: The Challenge of Feeding 9 Billion People”. In: *Science* 327.5967 (2010), pp. 812–818. doi: 10.1126/science.1185383.
- [44] P. Gole, P. Bedi, S. Marwaha, M. A. Haque, and C. K. Deb. “TrIncNet: a lightweight vision transformer network for identification of plant diseases”. In: *Frontiers in Plant Science* 14 (2023). doi: 10.3389/fpls.2023.1221557.
- [45] Google Inc. Keyhole Markup Language.
- [46] D. F. Gordon and M. Desjardins. “Evaluation and selection of biases in machine learning”. In: *Machine Learning* 20.1 (July 1, 1995), pp. 5–22. doi: 10.1007/BF00993472.
- [47] F. Görlich, E. Marks, A.-K. Mahlein, K. König, P. Lottes, and C. Stachniss. “UAV-Based Classification of Cercospora Leaf Spot Using RGB Images”. In: *Drones* 5.2 (2021). doi: 10.3390/drones5020034.
- [48] E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher. “Assessment and comparison of prognostic classification schemes for survival data”. In: *Statistics in Medicine* 18.17-18 (1999), pp. 2529–2545. doi: 10.1002/(SICI)1097-0258(19990915/30)18:17/18<2529::AID-SIM274>3.0.CO;2-5.
- [49] G. Grenzdörffer. “Automatic Generation Of Geometric Parameters Of Individual Cauliflower Plants For Rapid Phenotyping Using Dron Images”. In: *International Archives Of The Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).
- [50] E. Hamuda, M. Glavin, and E. Jones. “A survey of image processing techniques for plant extraction and segmentation in the field”. In: *Computers and Electronics in Agriculture* 125 (July 2016), pp. 184–199. doi: 10.1016/j.compag.2016.04.024.
- [51] F. E. Harrell Jr., K. L. Lee, and D. B. Mark. “Multivariate Multivariable Prognostic Models: Issues in Developing Models, Evaluating Assumptions and Adequacy, and Measuring and Reducing Errors”. In: *Statistics in Medicine* 15.4 (1996), pp. 361–387. doi: 10.1002/(SICI)1097-0258(19960229)15:4<361::AID-SIM168>3.0.CO;2-4.
- [52] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. 2015.
- [53] L. Held, N. Hens, P. O’Neill, and J. Wallinga, eds. Handbook of Infectious Disease Data Analysis. en. London, England: Chapman and Hall, Nov. 2019.
- [54] S. Hemalatha and J. J. B. Jayachandran. “A Multitask Learning-Based Vision Transformer for Plant Disease Localization and Classification”. In: *International Journal of Computational Intelligence Systems* 17.1 (July 18, 2024), p. 188. doi: 10.1007/s44196-024-00597-3.
- [55] H. W. Hethcote. “The Mathematics of Infectious Diseases”. In: *SIAM Review* 42.4 (2000), pp. 599–653. doi: 10.1137/S0036144500371907.
- [56] C. D. Holen and A. G. Dexter. “A Growing Degree Day Equation For Early Sugarbeet Leaf Stages”. In: *Sugarbeet Research and Extension Reports* 27 (1997), pp. 152–157.
- [57] D. W. Hosmer, S. Lemeshow, and S. May. Applied Survival Analysis: Regression Modeling of Time-to-Event Data. en. 2nd ed. Wiley Series in Probability and Statistics. Chichester, England: Wiley-Blackwell, Feb. 2008.

- [58] C. Huang and G. Asner. “Applications of Remote Sensing to Alien Invasive Plant Studies”. In: *Sensors* 9.6 (June 2009), pp. 4869–4889. doi: 10.3390/s90604869.
- [59] A. Huete, H. Liu, K. Batchily, and W. van Leeuwen. “A comparison of vegetation indices over a global set of TM images for EOS-MODIS”. In: *Remote Sensing of Environment* 59.3 (1997), pp. 440–451. doi: 10.1016/S0034-4257(96)00112-5.
- [60] P. Instruments. Sugar beet disease models. URL: <https://metos.at/en/disease-models-sugar-beet/> (visited on 07/31/2023).
- [61] S. Jain and B. C. Wallace. “Attention is not Explanation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 3543–3556. doi: 10.18653/v1/N19-1357.
- [62] K. James and K. Bradshaw. “Detecting plant species in the field with deep learning and drone technology”. In: *Methods in Ecology and Evolution* 11.11 (Sept. 2020). Ed. by S. McMahon, pp. 1509–1519. doi: 10.1111/2041-210x.13473.
- [63] S. Jay, A. Comar, R. Benicio, et al. “Scoring Cercospora Leaf Spot on Sugar Beet: Comparison of UGV and UAV Phenotyping Systems”. In: *Plant Phenomics* 2020 (Aug. 2020), pp. 1–18. doi: 10.34133/2020/9452123.
- [64] K. Jha, A. Doshi, P. Patel, and M. Shah. “A comprehensive review on automation in agriculture using artificial intelligence”. In: *Artificial Intelligence in Agriculture* 2 (2019), pp. 1–12. doi: 10.1016/j.aiia.2019.05.004.
- [65] J. M. Joyce. “Kullback-Leibler Divergence”. In: *International Encyclopedia of Statistical Science*. Ed. by M. Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. doi: 10.1007/978-3-642-04898-2_327.
- [66] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha. AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing. 2021. doi: 10.48550/ARXIV.2108.05542.
- [67] A. Kamilaris and F. Prenafeta-Boldú. “Deep learning in agriculture: A survey”. In: *Computers and Electronics in Agriculture* 147 (2018), pp. 70–90.
- [68] A. Kamilaris, A. Kartakoullis, and F. X. Prenafeta-Boldú. “A review on the practice of big data analysis in agriculture”. In: *Computers and Electronics in Agriculture* 143 (2017), pp. 23–37. doi: 10.1016/j.compag.2017.09.037.
- [69] A. Karpatne, G. Atluri, J. H. Faghmous, et al. “Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.10 (2017), pp. 2318–2331. doi: 10.1109/TKDE.2017.2720168.
- [70] S. M. Kazemi, R. Goel, S. Eghbali, et al. Time2Vec: Learning a Vector Representation of Time. 2019.
- [71] M. J. Keeling and P. Rohani. Modeling Infectious Diseases in Humans and Animals. Princeton University Press, 2008.
- [72] M. J. Keeling and P. Rohani. Modeling Infectious Diseases in Humans and Animals. Princeton University Press, 2008.
- [73] A. Kiniec, M. Spychalski, W. Miziniak, M. Palacz, and R. Kukawka. “The Use of Thyme (*Thymus vulgaris*) Essential Oil for Controlling Cercospora Leaf Spot (*Cercospora beticola*) on Sugar Beets (*Beta vulgaris*)”. In: *Agriculture* 14.11 (2024). doi: 10.3390/agriculture14112017.

- [74] J. P. Klein and M. L. Moeschberger. Survival Analysis Techniques for Censored and Truncated Data. Second. 2003.
- [75] Kleinwanzlebener Saatzucht AG, Rabbethge and Giesecke. Cercospora traits leaflet (*german: Cercospora-Klappkarte*). 1970.
- [76] M. Kulyal and P. Saxena. "Machine Learning approaches for Crop Yield Prediction: A Review". In: *2022 7th International Conference on Computing, Communication and Security (ICCCS)*. 2022, pp. 1–7. doi: 10.1109/ICCCS55188.2022.10079240.
- [77] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.
- [78] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis. "Machine Learning in Agriculture: A Review". In: *Sensors* 18.8 (2018). doi: 10.3390/s18082674.
- [79] T. Lin, Y. Wang, X. Liu, and X. Qiu. A Survey of Transformers. 2021.
- [80] D. B. Lobell. "The use of satellite data for crop yield gap analysis". In: *Field Crops Research* 143 (2013). Crop Yield Gap Analysis – Rationale, Methods and Applications, pp. 56–64. doi: 10.1016/j.fcr.2012.08.008.
- [81] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. 2019.
- [82] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss. "UAV-based crop and weed classification for smart farming". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 3024–3031. doi: 10.1109/ICRA.2017.7989347.
- [83] M. Louhaichi, M. M. Borman, and D. E. Johnson. "Spatially Located Platform and Aerial Photography for Documentation of Grazing Impacts on Wheat". In: *Geocarto International* 16.1 (2001), pp. 65–70. doi: 10.1080/10106040108542184.
- [84] D. J. Lovell, S. J. Powers, S. J. Welham, and S. R. Parker. "A perspective on the measurement of time in plant disease epidemiology". In: *Plant Pathology* 53.6 (2004), pp. 705–712. doi: 10.1111/j.1365-3059.2004.01097.x.
- [85] T. Luong, H. Pham, and C. D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Ed. by L. Màrquez, C. Callison-Burch, and J. Su. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. doi: 10.18653/v1/D15-1166.
- [86] L. V. Madden, G. Hughes, and F. van den Bosch. The Study of Plant Disease Epidemics. The American Phytopathological Society, 2017. doi: 10.1094/9780890545058.
- [87] W. H. Maes and K. Steppe. "Perspectives for Remote Sensing with Unmanned Aerial Vehicles in Precision Agriculture". In: *Trends in Plant Science* 24.2 (2019), pp. 152–164. doi: 10.1016/j.tplants.2018.11.007.
- [88] A.-K. Mahlein, M. Kuska, J. Behmann, G. Polder, and A. Walter. "Hyperspectral Sensors and Imaging Technologies in Phytopathology: State of the Art". In: *Annual Review of Phytopathology* 56.1 (2018). PMID: 30149790, pp. 535–558. doi: 10.1146/annurev-phyto-080417-050100.
- [89] A.-K. Mahlein, T. Rumpf, P. Welke, et al. "Development of spectral indices for detecting and identifying plant diseases". In: *Remote Sensing of Environment* 128 (2013), pp. 21–30. doi: 10.1016/j.rse.2012.09.019.
- [90] A.-K. Mahlein. "Plant Disease Detection by Imaging Sensors – Parallels and Specific Demands for Precision Agriculture and Plant Phenotyping". In: *Plant Disease* 100.2 (2016). PMID: 30694129, pp. 241–251. doi: 10.1094/PDIS-03-15-0340-FE.

- [91] A. McBratney, B. Whelan, T. Ancev, and J. Bouma. “Future Directions of Precision Agriculture”. In: *Precision Agriculture* 6.1 (Feb. 1, 2005), pp. 7–23. doi: 10.1007/s11119-005-0681-8.
- [92] M. B. McDermott, H. Zhang, L. H. Hansen, G. Angelotti, and J. Gallifant. “A closer look at AUROC and AUPRC under class imbalance”. In: *Proceedings of the 38th International Conference on Neural Information Processing Systems*. NIPS ’24. Vancouver, BC, Canada: Curran Associates Inc., 2024.
- [93] I. Mehedi, M. S. Hanif, M. Bilal, M. Vellingiri, and T. Palaniswamy. “Remote Sensing and Decision Support System Applications in Precision Agriculture: Challenges and Possibilities”. In: *IEEE Access* PP (Jan. 2024). doi: 10.1109/ACCESS.2024.3380830.
- [94] U. Meier. *Entwicklungsstadien mono- und dikotyler Pflanzen: BBCH Monografie*. 2018.
- [95] P. Michel, O. Levy, and G. Neubig. “Are Sixteen Heads Really Better than One?” In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.
- [96] M. Moran, Y. Inoue, and E. Barnes. “Opportunities and limitations for image-based remote sensing in precision crop management”. In: *Remote Sensing of Environment* 61.3 (1997), pp. 319–346. doi: 10.1016/S0034-4257(97)00045-X.
- [97] D. J. Mulla. “Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps”. In: *Biosystems Engineering* 114.4 (2013). Special Issue: Sensing Technologies for Sustainable Agriculture, pp. 358–371. doi: 10.1016/j.biosystemseng.2012.08.009.
- [98] A. Myronenko and X. Song. “Point Set Registration: Coherent Point Drift”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 32.12 (Dec. 2010), pp. 2262–2275. doi: 10.1109/TPAMI.2010.46.
- [99] V. Nedungadi, X. Xiong, A. Potze, et al. *From General to Specialized: The Need for Foundational Models in Agriculture*. 2025. doi: 10.48550/arXiv.2507.05390.
- [100] H. Niu and Y. Chen, eds. *Smart Big Data in Digital Agriculture Applications: Advances in Digital Agriculture: Emerging Technologies and Applications*. Agriculture Automation and Control. Cham, Switzerland: Springer, 2024.
- [101] H. Niu and Y. Chen. “The Unmanned Ground Vehicles (UGVs) for Digital Agriculture”. In: *Smart Big Data in Digital Agriculture Applications*. Ed. by H. Niu and Y. Chen. Agriculture Automation and Control. Cham, Switzerland: Springer, 2024, pp. 99–109. doi: 10.1007/978-3-031-52645-9_5.
- [102] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.
- [103] North Dakota Agricultural Weather Network. Sugarbeet Growth Stage Development and Growing Degree Day Accumulation. URL: <https://ndawn.ndsu.nodak.edu/help-sugarbeet-growing-degree-days.html> (visited on 10/22/2024).
- [104] F. Nutter Jr, P. Teng, and F. Shokes. “Disease assessment terms and concepts”. In: *Plant Disease* 75 (Jan. 1991), pp. 1187–1188.
- [105] D. S. de O. Santos Júnior, P. S. de Mattos Neto, J. F. de Oliveira, and G. D. Cavalcanti. “A hybrid system based on ensemble learning to model residuals for time series forecasting”. In: *Information Sciences* 649 (2023), p. 119614. doi: 10.1016/j.ins.2023.119614.
- [106] A. Oikonomidis, C. Catal, and A. Kassahun. “Deep learning for crop yield prediction: a systematic literature review”. In: *New Zealand Journal of Crop and Horticultural Science* 51.1 (2023), pp. 1–26. doi: 10.1080/01140671.2022.2032213.

- [107] V.-E. Oniga, A.-I. Breaban, and F. Statescu. “Determining the Optimum Number of Ground Control Points for Obtaining High Precision Results Based on UAS Images”. In: *Proceedings 2.7* (2018). doi: 10.3390/ecrs-2-05165.
- [108] OPENGIS.ch. QField - A Free and Open Source Geographic Information System.
- [109] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. doi: 10.1109/TSMC.1979.4310076.
- [110] O. Patel, Y. P. S. Maravi, and S. Sharma. “A Comparative Study of Histogram Equalization Based Image Enhancement Techniques for Brightness Preservation and Contrast Enhancement”. In: *CoRR abs/1311.4033* (2013).
- [111] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.
- [112] F. J. Pierce and P. Nowak. “Aspects of Precision Agriculture”. In: ed. by D. L. Sparks. Vol. 67. *Advances in Agronomy*. Academic Press, 1999, pp. 1–85. doi: 10.1016/S0065-2113(08)60513-1.
- [113] S. M. Pizer, E. P. Amburn, J. D. Austin, et al. “Adaptive histogram equalization and its variations”. In: *Computer Vision, Graphics, and Image Processing* 39.3 (1987), pp. 355–368. doi: 10.1016/S0734-189X(87)80186-X.
- [114] QGIS Development Team. QGIS Geographic Information System. Open Source Geospatial Foundation. 2009.
- [115] P. Racca and E. Jörg. “CERC BET 3 – a forecaster for epidemic development of *Cercospora beticola*”. In: *EPPO Bulletin* 37.2 (2007), pp. 344–349. doi: 10.1111/j.1365-2338.2007.01136.x.
- [116] A. E. Raftery. “A Model for High-Order Markov Chains”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 47.3 (Dec. 2018), pp. 528–539. doi: 10.1111/j.2517-6161.1985.tb01383.x.
- [117] M. Raissi, P. Perdikaris, and G. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. doi: 10.1016/j.jcp.2018.10.045.
- [118] R. Raj, S. Kar, R. Nandan, and A. Jagarlapudi. “Precision Agriculture and Unmanned Aerial Vehicles (UAVs)”. In: *Unmanned Aerial Vehicle: Applications in Agriculture and Environment*. Ed. by R. Raj and W. Teiji. Springer, 2020, pp. 7–23. doi: 10.1007/978-3-030-27157-2_2.
- [119] L. I. Rangel, R. E. Spanner, M. K. Ebert, et al. “*Cercospora beticola*: The intoxicating lifestyle of the leaf spot pathogen of sugar beet”. In: *Molecular Plant Pathology* 21.8 (2020), pp. 1020–1041. doi: 10.1111/mpp.12962.
- [120] M. Reichstein, G. Camps-Valls, B. Stevens, et al. “Deep learning and process understanding for data-driven Earth system science”. In: *Nature* 566.7743 (Feb. 1, 2019), pp. 195–204. doi: 10.1038/s41586-019-0912-1.
- [121] B. U. Rekha, V. V. Desai, P. S. Ajawan, and S. K. Jha. “Remote Sensing Technology and Applications in Agriculture”. In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. 2018, pp. 193–197. doi: 10.1109/CTEMS.2018.8769124.

- [122] G. Rondeaux, M. Steven, and F. Baret. "Optimization of soil-adjusted vegetation indices". In: *Remote Sensing of Environment* 55.2 (1996), pp. 95–107. doi: 10.1016/0034-4257(95)00186-7.
- [123] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Cham: Springer International Publishing, Nov. 18, 2015, pp. 234–241.
- [124] V. Rossi, P. Racca, and P. Battilani. "A simulation model for Cercospora leaf spot epidemics on sugarbeet". In: *Phytopathologia Mediterranea* 33.2 (1994), pp. 105–112.
- [125] L. von Rueden, S. Mayer, K. Beckh, et al. "Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems". In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2023), pp. 614–633. doi: 10.1109 / TKDE . 2021 . 3079836.
- [126] O. Russakovsky, J. Deng, H. Su, et al. ImageNet Large Scale Visual Recognition Challenge. 2015.
- [127] P. A. Saccardo. "Fungi Veneti novi vel critici v. mycologiae Veneti addendi. Series XII". In: *Michelia* 2 (1881), pp. 241–301.
- [128] F. Savian, M. Martini, P. Ermacora, S. Paulus, and A.-K. Mahlein. "Prediction of the Kiwifruit Decline Syndrome in Diseased Orchards by Remote Sensing". In: *Remote Sensing* 12.14 (2020). doi: 10.3390/rs12142194.
- [129] D. Schimmelpfennig. Farm Profits and Adoption of Precision Agriculture. Economic Research Report 249773. United States Department of Agriculture, Economic Research Service, Oct. 2016. doi: 10.22004/ag.econ.249773.
- [130] S. Serrano and N. A. Smith. "Is Attention Interpretable?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by A. Korhonen, D. Traum, and L. Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2931–2951. doi: 10.18653/v1/P19-1282.
- [131] M. Shahhosseini, G. Hu, I. Huber, and S. V. Archontoulis. "Coupling machine learning and crop modeling improves crop yield prediction in the US Corn Belt". In: *Scientific Reports* 11.1 (Jan. 15, 2021), p. 1606. doi: 10.1038/s41598-020-80820-1.
- [132] W. W. Shane. "Impact of Cercospora Leaf Spot on Root Weight, Sugar Yield, and Purity of Beta vulgaris". In: *Plant Disease* 76.8 (1992), p. 812. doi: 10.1094/pd-76-0812.
- [133] T. Shi, Y. Liu, X. Zheng, et al. "Recent advances in plant disease severity assessment using convolutional neural networks". In: *Scientific Reports* 13.1 (Feb. 9, 2023), p. 2336. doi: 10.1038/s41598-023-29230-7.
- [134] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015.
- [135] J. P. Snyder. Map projections: A working manual. Report. Washington, D.C., 1987. doi: 10.3133/pp1395.
- [136] Supriya, A. Shukla, and M. Manchanda. "A Survey Paper on Precision Agriculture based Intelligent system for Plant Leaf Disease Identification". In: *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)*. 2023, pp. 1–6. doi: 10.1109/ICAIA57370.2023.10169852.
- [137] K. Suresh, C. Severn, and D. Ghosh. "Survival prediction models: an introduction to discrete-time modeling". In: *BMC Medical Research Methodology* 22.1 (July 26, 2022), p. 207. doi: 10.1186/s12874-022-01679-6.

- [138] D. Tapete, G. Lazzeri, J. A. G. Garcia, et al. "Earth observation hyperspectral remote sensing in agriculture: capacity building experience within the Horizon Europe EXPERT project between ASI, DLR, and ROSA space agencies". In: *Remote Sensing for Agriculture, Ecosystems, and Hydrology XXVII*. Ed. by C. M. U. Neale, A. Maltese, C. R. B. Jr., and A. Castagna. Vol. 13666. International Society for Optics and Photonics. SPIE, 2025, 136660H. doi: 10.1117/12.3069834.
- [139] M. U. Tariq, S. M. Saqib, T. Mazhar, M. A. Khan, T. Shahzad, and H. Hamam. "Edge-enabled smart agriculture framework: Integrating IoT, lightweight deep learning, and agentic AI for context-aware farming". In: *Results in Engineering* 28 (2025), p. 107342. doi: 10.1016/j.rineng.2025.107342.
- [140] A. Thenkabail, P. Lyon, and J. Huete. Hyperspectral remote sensing of vegetation. en. Ed. by P. S. Thenkabail, J. G. Lyon, and A. Huete. Boca Raton, FL: CRC Press, Apr. 2016.
- [141] C. J. Tucker. "Red and photographic infrared linear combinations for monitoring vegetation". In: *Remote Sensing of Environment* 8.2 (1979), pp. 127–150. doi: 10.1016/0034-4257(79)90013-0.
- [142] J. Valente, B. Sari, L. Kooistra, H. Kramer, and S. Mùcher. "Automated crop plant counting from very high-resolution aerial imagery". In: *Precision Agriculture* 21.6 (May 2020), pp. 1366–1384. doi: 10.1007/s11119-020-09725-3.
- [143] T. van Klompenburg, A. Kassahun, and C. Catal. "Crop yield prediction using machine learning: A systematic literature review". In: *Computers and Electronics in Agriculture* 177 (2020), p. 105709. doi: 10.1016/j.compag.2020.105709.
- [144] A. Vaswani, N. Shazeer, N. Parmar, et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. Vol. 30. Curran Associates, Inc., 2017.
- [145] J. Vereijssen, J. H. M. Schneider, A. Stein, and M. J. Jeger. "Spatial pattern of Cercospora leaf spot of sugar beet in fields in long- and recently-established areas". In: *European Journal of Plant Pathology* 116.3 (Nov. 2006), pp. 187–198. doi: 10.1007/s10658-006-9046-z.
- [146] J. Vig. "A Multiscale Visualization of Attention in the Transformer Model". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Ed. by M. R. Costa-jussà and E. Alfonseca. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 37–42. doi: 10.18653/v1/P19-3007.
- [147] G. Wang, Y. Sun, and J. Wang. "Automatic Image-Based Plant Disease Severity Estimation Using Deep Learning". In: *Computational Intelligence and Neuroscience* 2017 (July 2017), pp. 1–8. doi: 10.1155/2017/2917536.
- [148] J. Weiland and G. Koch. "Sugarbeet leaf spot disease (*Cercospora beticola* Sacc.)†". In: *Molecular Plant Pathology* 5.3 (2004), pp. 157–166. doi: 10.1111/j.1364-3703.2004.00218.x.
- [149] M. Weiss, F. Jacob, and G. Duveiller. "Remote sensing for agricultural applications: A meta-review". In: *Remote Sensing of Environment* 236 (2020), p. 111402. doi: 10.1016/j.rse.2019.111402.
- [150] S. Wiegrefe and Y. Pinter. "Attention is not not Explanation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by K. Inui, J. Jiang, V. Ng, and X. Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 11–20. doi: 10.18653/v1/D19-1002.

- [151] D. Wilks. Statistical Methods in the Atmospheric Sciences: An Introduction. International geophysics series. Academic Press, 1995.
- [152] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems. 2022.
- [153] J. D. Willard, X. Jia, S. Xu, M. S. Steinbach, and V. Kumar. “Integrating Physics-Based Modeling with Machine Learning: A Survey”. In: *ArXiv abs/2003.04919* (2020).
- [154] C. E. Windels, H. A. Lamey, D. Hilde, J. Widner, and T. Knudsen. “A Cerospora Leaf Spot Model for Sugar Beet: In Practice by an Industry”. In: *Plant Disease* 82.7 (1998). PMID: 30856938, pp. 716–726. doi: 10.1094/PDIS.1998.82.7.716.
- [155] P. F. J. Wolf, M. Heindl, and J. A. Verreet. “Zum Einfluss des Bestandesklimas auf die Prädisposition der Zuckerrübe gegenüber Infektionen von *Cercospora beticola* (Sacc.) / Influence of sugar beet leaf mass development on predisposition of the crop to *Cercospora beticola* (Sacc.)” In: *Zeitschrift für Pflanzenkrankheiten und Pflanzenschutz / Journal of Plant Diseases and Protection* 108.6 (2001), pp. 578–592.
- [156] P. F. J. Wolf and J. A. Verreet. “An Integrated Pest Management System in Germany for the Control of Fungal Leaf Diseases in Sugar Beet: The IPM Sugar Beet Model”. In: *Plant Disease* 86.4 (Apr. 2002), pp. 336–344. doi: 10.1094/pdis.2002.86.4.336.
- [157] P. F. J. Wolf and J. A. Verreet. “Factors Affecting the Onset of *Cercospora* Leaf Spot Epidemics in Sugar Beet and Establishment of Disease-Monitoring Thresholds”. In: *Phytopathology* 95.3 (2005). PMID: 18943120, pp. 269–274. doi: 10.1094/PHYTO-95-0269.
- [158] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt. “Big Data in Smart Farming – A review”. In: *Agricultural Systems* 153 (2017), pp. 69–80. doi: 10.1016/j.agsy.2017.01.023.
- [159] D. Xu, Y. Lu, H. Liang, Z. Lu, L. Yu, and Q. Liu. “Areca Yellow Leaf Disease Severity Monitoring Using UAV-Based Multispectral and Thermal Infrared Imagery”. In: *Remote Sensing* 15.12 (2023). doi: 10.3390/rs15123114.
- [160] R. Xu, C. Li, and A. H. Paterson. “Multispectral imaging and unmanned aerial systems for cotton plant phenotyping”. In: *PLOS ONE* 14.2 (Feb. 2019), pp. 1–20. doi: 10.1371/journal.pone.0205083.
- [161] J. Xue and B. Su. “Significant Remote Sensing Vegetation Indices: A Review of Developments and Applications”. In: *Journal of Sensors* 2017 (2017), pp. 1–17. doi: 10.1155/2017/1353691.
- [162] S. Yin, Y. Xi, X. Zhang, C. Sun, and Q. Mao. “Foundation Models in Agriculture: A Comprehensive Review”. In: *Agriculture* 15.8 (2025). doi: 10.3390/agriculture15080847.
- [163] P. Yu, F. Teng, W. Zhu, C. Shen, Z. Chen, and J. Song. “Cloud–edge–device collaborative computing in smart agriculture: architectures, applications, and future perspectives”. In: *Frontiers in Plant Science* Volume 16 - 2025 (2025). doi: 10.3389/fpls.2025.1668545.
- [164] J. C. Zadoks, T. T. Chang, and C. F. Konzak. “A decimal code for the growth stages of cereals”. In: *Weed Research* 14.6 (1974), pp. 415–421. doi: 10.1111/j.1365-3180.1974.tb01084.x.
- [165] C. Zhang and J. M. Kovacs. “The application of small unmanned aerial systems for precision agriculture: a review”. In: *Precision Agriculture* 13.6 (Dec. 1, 2012), pp. 693–712. doi: 10.1007/s11119-012-9274-5.
- [166] N. Zhang, M. Wang, and N. Wang. “Precision agriculture—a worldwide overview”. In: *Computers and Electronics in Agriculture* 36.2 (2002), pp. 113–132. doi: 10.1016/S0168-1699(02)00096-0.

- [167] Z. H. Zul Azlan, S. N. Junaini, N. A. Bolhassan, R. Wahi, and M. A. Arip. “Harvesting a sustainable future: An overview of smart agriculture’s role in social, economic, and environmental sustainability”. In: *Journal of Cleaner Production* 434 (2024), p. 140338. doi: 10.1016/j.jclepro.2023.140338.

Danksagung

Abschließend möchte ich die Gelegenheit ergreifen, einigen Menschen zu danken, die mich während der Anfertigung dieser Dissertation unterstützt haben und die Forschung hinter dieser Arbeit erst ermöglicht haben.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Christian Bauckhage für die Betreuung dieser Arbeit, die inspirierenden Diskussionen und die Möglichkeit, in seiner Arbeitsgruppe forschen und promovieren zu dürfen. Ebenso danke ich Prof. Dr. Stefan Wrobel für die Übernahme des Zweitgutachtens.

Ein großer Dank geht an das Sensorik-Team des Instituts für Zuckerrübenforschung (IfZ) in Göttingen für die großartige Zusammenarbeit. Insbesondere an Prof. Dr. Anne-Katrin Mahlein und Facundo Ispizua für die enge Kollaboration bei allen wesentlichen Forschungsarbeiten dieser Dissertation. Ohne Facundos Unterstützung und den fachübergreifenden Austausch mit ihm wäre diese Arbeit nicht möglich gewesen. Aus unserer Zusammenarbeit ist nicht nur, wie ich finde, eine tolle wissenschaftliche Arbeit entstanden, sondern auch eine Freundschaft, die ich sehr schätze und die hoffentlich noch lange bestehen bleibt.

Das Exzellenzcluster PhenoRob sowie das Lamarr-Institut, deren Teil ich während meiner Promotion sein durfte, haben mir eine hervorragende Forschungsumgebung geboten. Der interdisziplinäre Austausch und die familiäre Atmosphäre während zahlreicher Veranstaltungen waren stets angenehm und haben mich in meiner wissenschaftlichen Arbeit sehr unterstützt. Dafür möchte ich mich herzlich bedanken.

Mein Dank gilt auch meinen tollen Kolleg*innen am Institut für Informatik III der Universität Bonn für die angenehme Arbeitsatmosphäre und die vielen inspirierenden Gespräche. Ebenso danke ich meinen tollen Kolleg*innen am Fraunhofer IAIS in Sankt Augustin, die mich während meiner Promotion unterstützt und mir stets mit Rat und Tat zur Seite gestanden haben. Ein besonderer Dank soll an dieser Stelle an meine Teamleiterin Sandra Halscheidt, die mich bei der Promotion sehr unterstützt und mir neben der Arbeit an anderen Projekten immer wieder Freiräume für die Forschung an dieser Arbeit ermöglicht hat.

Zu guter Letzt geht mein Dank an meine Freundin Anna-Maria, meine Familie und meine Freundinnen und Freunde, die mich während der gesamten Zeit der Promotion begleitet, motiviert und unterstützt haben. Ohne euch wäre diese Arbeit nicht möglich gewesen. Danke, dass ihr immer für mich da seid!