# MASTERARBEIT | MASTER'S THESIS

Titel | Title

Artificial Intelligence for Airborne Phenotyping

verfasst von | submitted by

Lorenzo Beltrame

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

| | |
|---|---|
| Studienkennzahl lt. Studienblatt \| Degree programme code as it appears on the student record sheet: | UA 066 645 |
| Studienrichtung lt. Studienblatt \| Degree programme as it appears on the student record sheet: | Masterstudium Data Science |
| Betreut von \| Supervisor: | Assoz. Prof. Dipl.-Ing. Dr.techn. Sebastian Tschiatschek BSc |

**Abstract**

This thesis explores the application of machine learning (ML) techniques for phenotyping using multispectral and multitemporal airborne data, focusing on the identification of yellow rust in wheat. The primary aim is to evaluate the efficacy of aerial ML-based phenotyping as an alternative to traditional in situ methods. A pioneering aspect of this study is the creation of a novel dataset comprising time series multispectral images, each depicting an experimental plot within a wheat field, alongside corresponding yellow rust disease scores provided by experts. A comparative analysis is then conducted between various basic ML models and deep learning models to predict yellow rust using the dataset. Our findings reveal the challenges faced by basic ML models in accurately predicting yellow rust, contrasting with the promising results achieved by a deep learning model utilising the ResNet34 feature extractor. These results underscore the potential of ML approaches in remote phenotyping for plant breeding, particularly when integrating deep learning models with attention mechanisms. The study provides valuable insights into remote phenotyping techniques, with implications for enhancing disease monitoring and crop management practices. However, further refinement is necessary. The observed lower performance on one of the test sets suggests a scarcity of data essential for ensuring model robustness and generalisation capabilities. Future research could explore additional spectral indices and automated spectral band selection. Finally, with more data, the model could also be trained to be more robust to arbitrary time steps, a critical property for its practical usability.

## Acknowledgments

I wish to express my sincere gratitude to Sebastian Tschiatschek for his invaluable guidance and support as my official supervisor at the University of Vienna. His mentorship has been pivotal in shaping my academic journey.

Additionally, I am deeply thankful to Phillipp Fanta-Jende for his inspiring supervision during my time at AIT and for his significant contribution to overseeing this thesis. His expertise and encouragement have greatly enriched my learning experience and facilitated my growth in the field.

Furthermore, I would like to extend my appreciation to Jules Salzinger for his dedicated supervision and rigorous teaching approach. His mentorship has been instrumental in nurturing both my analytical and technical skills.

I would also like to thank all my colleagues from the AAS unit for their support and collaboration.

Finally, I want to express my gratitude to all my classmates from the Data Science Master's program for their unwavering support over the past months, particularly for their assistance in reviewing this manuscript.

# Contents

# 1 Introduction

The issue of global food security is becoming increasingly pressing as the world's population is projected to double its demand for crop production by 2050, as reported by Ray et al. 2013. However, the agricultural industry faces a formidable challenge to meet this surge in demand. As reported by Ray et al. 2012, crop yields would need to increase at 2.4% per year, exceeding the current average increase rate of 1.3%. Moreover, up to 40% of land under cereal production has experienced stagnant yields and climate change intensifies severe drought phenomena and flooding. Despite significant advances in breeding and agronomy over the past 50 years, achieving such an ambitious goal remains an uphill battle.

Fortunately, as Araus and Cairns 2014 suggests, machine learning and remote sensing techniques may provide valuable assistance in tackling this complex challenge. With their ability to acquire and analyse vast amounts of data, these technologies can provide critical and accurate insight into the state of crops, as illustrated in Gracia-Romero et al. 2019. This, in turn, can enable more effective and targeted interventions, such as precision agriculture (PA), which can contribute to increasing crop yields. Those technologies can also provide another key insight: phenotype trait prediction. This becomes particularly valuable since ongoing advances in breeding techniques offer the potential to significantly accelerate crops' genetic improvement rate. In particular, remote sensing and machine learning now allow breeders to select the best crops, while research on the genetics of quantitative traits, such as yield, water stress, and resistance to fungi, has proven highly effective, as demonstrated in previous studies such as Gracia-Romero et al. 2019 and Oerke 2020. These breakthroughs in breeding and genetics have the potential to transform the agricultural industry by enabling breeders to develop crops that are better adapted to challenging growing conditions and more efficient in their use of resources. Therefore, these tools hold great potential in tackling the pressing challenge of global food security amidst the impact of climate change.

## 1.1 Project Description

In this particular context, the Lower Austrian state government initiated a project to enhance the yield of Winter Wheat. The project, spanning three years, involves technical assistance from the Austrian Institute of Technology (AIT) and the expertise of Saatzuch Edelhof, a long-standing private entity renowned for conducting field breeding experiments. The primary goal of the project is to identify suitable genetic material for the field experiments planned in the second and third years. To achieve this objective the various research entities have different roles.

Saatzuch Edelhof oversees soil preparation, seed preparation, planting, fertilisation, irrigation, and harvesting during field experiments, as well as recording weather data via mobile weather stations.

The AIT provides drone technology to enable fast collection of RGB, multispectral, and thermal image data over large-scale fields. Airborne data is acquired every two weeks for three years during the growing season, depending on the weather. Furthermore, the AIT offers data processing pipelines and predictive tools to automate, improve, and expedite the process of predicting phenotypic traits during field experiments. The overall role is to define and create a standardised way of acquiring and processing agricultural time-series data.

In this framework, the thesis comprises two distinct parts, partially covering one project work package. In the first part, field data, specifically airborne multispectral and Red, green, and blue (RGB) band data will be prepared for machine learning analysis. The second part involves using statistical methods to analyse the refined field data to unveil fundamental relationships and dependencies. One machine learning pipeline is deployed in the subsequent stages. This pipeline will predict a phenotypic trait: the yellow rust disease scores. Experimental activities will be carried out in Obersiebenbrunn, focusing on the study of 1062 field lines of winter wheat, as illustrated in Figure 1.

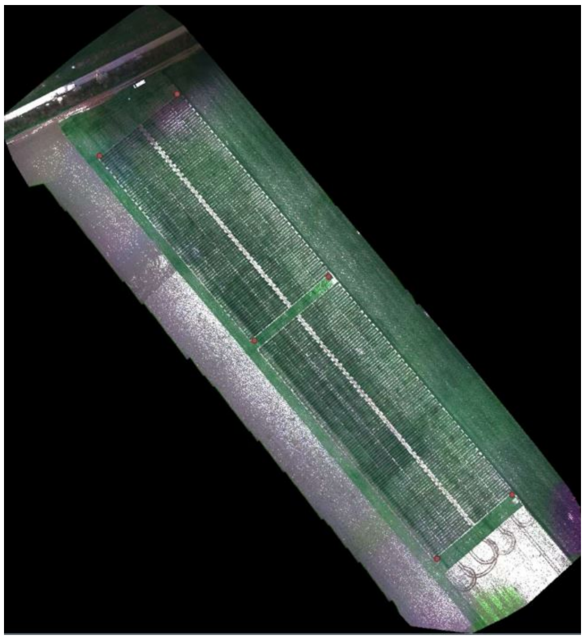


Figure 1: RGB-encoded orthorectified image mosaic of the Edelhof facility obtained with Pix4D.

## 1.2 Fungi Infection and Disease Score

The phenotype under investigation in this thesis is the disease score, which is related to the health assessment of winter wheat crops affected by yellow rust (*Puccinia striiformis*, also known as stripe rust). Disease scoring involves a thorough visual inspection, meticulously examining parameters such as the colour, texture, and size of leaves and stems.

This crucial task is carried out by the professionals stationed at the Edelhof facility. The results are translated into a disease score scale that ranges from 1 to 9. This scoring system offers a comprehensive representation of the extent of disease infestation within the plants. A score of 1 signifies plants unaffected by any biotic stress, while a score of 9 corresponds to plants experiencing severe disease symptoms. A detailed explanation can be found in Appendix A. It's important to note that distinct disease scores can be assigned to different plant diseases, tailoring the assessment to the specific context.

In this particular study, the disease scoring system chosen adheres to the Austrian standard and is advocated by the AGES (Austrian Agency for Health and Food Safety). Detailed information on this scoring system can be found in the Appendix (see Appendix A), providing the reader with a comprehensive understanding of the evaluation methodology and criteria. This comprehensive approach to disease scoring ensures a well-rounded analysis of the impact of yellow rust on winter wheat crops and serves as a foundation for the learning objectives of this thesis.

The ability to visually recognise the distinctive pustules of yellow rust on winter wheat leaves suggests the potential applicability of machine learning models for the same task. These recognisable features visible in the RGB bands provide valuable data that can be used to train algorithms to autonomously identify and classify disease symptoms accurately (Su et al. 2021, Mi et al. 2020). By bridging traditional visual assessment with modern machine learning, there is an opportunity to enhance disease detection efficiency and objectivity, advancing crop management practices and boosting operational speed.

Moreover, the use of multispectral data adds another layer of potential to the detection of diseases in crops, as proved by Franke and Menz 2007. Multispectral imaging captures information beyond the visible spectrum, enabling the differentiation of subtle variations in plant health and disease development that might not be discernible to the naked eye.

## 1.3 Research Question

The main objective of this thesis is to investigate the viability of ML-based aerial phenotyping as a substitute for traditional in situ data phenotyping for disease scores. The research question that this thesis addresses is:

*"To what extent is aerial ML-based phenotyping a viable alternative to traditional in situ phenotyping?"*

This research question holds significant importance, considering the potential of machine learning to support agronomists in measuring resource-intensive and time-consuming phenotypic traits. The emphasis of this thesis lies in introducing a novel element to the airborne phenotyping task—time series analysis. To narrow the scope, the investigation focuses on a single phenotype: the yellow rust disease score.

To comprehensively address the primary research question, it is essential to dive into various interconnected subquestions. These subquestions encompass the exploration of optimal techniques for preparing raw remote sensing data, the identification of effective algorithms for accurate disease scoring prediction, a study on the general capabilities to scale a phenotyping approach and how to convey temporal information into a machine learning model.

*"How can remote sensing image data be transformed and aggregated to extract characteristics for predicting disease scores?"*

The effective handling of raw drone-acquired data is a crucial consideration, laying the groundwork for the development of a robust machine learning model to address the primary research question. AIT's drones, equipped with an Altum camera, capture thermal and multispectral images at an altitude of 60 meters, resulting in a substantial dataset of approximately 50 GB per acquisition. These images span various bands, including RGB, panchromatic, Long Wave Infrared (LWIR), and Near Infrared (NIR).

Processing this voluminous dataset involves indispensable steps such as radiometric calibration, orthonormalisation, rectification, and translation into a structured format suitable for machine learning. Moreover, taking advantage of the resulting raster structure, the normalised vegetation index (NDVI) is a crucial parameter to be computed (Wójtowicz et al. 2016). Managing photogrammetric processing is streamlined with Pix4D, a commercially available tool adept at handling high-resolution, large-scale images suitable for machine learning pipelines (Rasmussen et al. 2016).

Leveraging Pix4D facilitates the acquisition of a set of geographically coherent reflectance maps. An illustrative example of an aerial image obtained through Pix4D is depicted in Figure 1.

Conversely, the latter part of this investigation entails the use of QGIS and the Pillow Python library to efficiently manage and visualise georeferenced data. This process includes associating a raster with each experimental plot, leading to the extraction of sub-images from the reflectance maps. These sub-images contribute to the creation of a geographically coherent stack of multispectral maps, commonly referred to as multispectral cubes.

This subquestion stands as a foundational centrepiece for aerial machine learning-based phenotyping, enabling the precise extraction of essential characteristics from aerial data. The incorporation of advanced techniques and tools enhances the viability of aerial-based phenotyping as a potential alternative to labour-intensive traditional in situ methods.

*"To what extent can predictions of yellow rust scores be made effectively using images acquired by mid-altitude UAVs, compared to data obtained from both low-altitude UAV and handheld image captures?"*

This subquestion is directly aligned with our primary research objective, focusing on a crucial aspect of the viability of aerial machine learning (ML)-based phenotyping. The choice of unmanned aerial vehicle (UAV) altitude is pivotal in gauging the practicality and effectiveness of our approach. The current body of literature encompasses diverse approaches for wheat plant phenotyping, ranging in complexity. Notably, studies such as Mi et al. 2020 and Koc et al. 2022 employ on-site image data from field-level tools, showcasing the effectiveness of machine learning methods in addressing predictive challenges.

Conversely, other methodologies rely on low-altitude UAV flights. For instance, Zhang et al. 2019 utilise hyperspectral data collected at a low altitude of 30 meters to successfully predict the presence of yellow rust. Our contribution aims to emphasise the potential for yellow rust prediction through flights at an altitude of 60 meters. Although challenging, the use of pre-trained image models, such as ResNet, could tackle this challenge effectively. In fact, the initial layers of this model excel at detecting low-resolution patterns, which is particularly relevant for higher-altitude flights. If successful, this approach could provide a valuable tool to bridge the gap and scale our methodology to an industrial level, enabling a faster and less data-burdensome approach to large-scale yellow rust prediction.

*"How can we effectively predict phenotypic traits in crops, particularly focusing on disease scoring, considering the trade-off between model simplicity and complexity?"*

In the pursuit of precise prediction of phenotypic traits in wheat, with a specific emphasis on disease scoring, a critical focus is on striking a delicate balance between model simplicity and complexity. In alignment with an iterative framework advocated in the literature (Virnodkar et al. 2020, Singh et al. 2018, Nguyen et al. 2023), our research is geared towards establishing a systematic approach. We initiate the process by constructing foundational models utilising well-established machine learning techniques, specifically linear regression, Support Vector Machines (SVMs) and Random Forests (RFs), aiming to create a robust baseline. This foundational framework, anchored in the recognised efficacy of simpler models for distinct facets of phenotypic trait prediction, is further reinforced by insights from Nguyen (Nguyen et al. 2023), endorsing the practicality of SVM and RF models in crop phenotyping. Significantly, we emphasise a comprehensive evaluation of the performance of these basic models against deep learning methods, particularly Convolutional Neural Networks (CNNs). This research approach underscores our commitment to discerning the relative efficacy of conventional and advanced methodologies, contributing substantially to the ongoing discourse in the field of crop science.

*"In the context of evaluating the viability of aerial machine learning-based phenotyping as an alternative to traditional in situ phenotyping, how can a machine learning model be designed to validate its performance, leveraging domain-specific agricultural knowledge?"*

For robust generalisation in machine learning models, evaluating the significance of internal feature maps is crucial, especially when considering the temporal dimension. In our latest deep learning architecture, we introduce an attention mechanism, inspired by neuroscience concepts (Zhao et al. 2018), to address this challenge. This mechanism computes a weighted sum of time and space feature maps. Additionally, we incorporate sparse L1-norm regularisation to penalize the model based on these weighted sums, optimising channel weights and mitigating overfitting. This reduction in features not only enhances model training efficiency but also acts as a strategic defense against overfitting. The attention mechanism enables adaptive selection of influential input features, providing a dynamic framework for testing domain-specific hypotheses. This approach empowers machine learning practitioners and agronomists to systematically explore the decision-making process within the model by granting autonomy to determine the relative importance of spatial and temporal components.

## 1.4 Contribution

Our thesis aims to explore the potential advantages of incorporating multispectral information for yellow rust characterisation using UAV-based multispectral images.

What sets our study apart is the acquisition of expert assessment ground truth data during Edelhof's scoring campaign, a methodologically crucial element that is both challenging to obtain and expensive to carry out.

The usage of ground control points significantly enhances geolocalisation precision. This precision enables us to precisely select the portion of the field where crops are growing without the need for a preliminary NDVI analysis to differentiate between soil and plant areas, as done in the study by Zhang et al. 2019.

Adding to this, our research introduces a novel element with a comprehensive dataset spanning the entire growth cycle, captured at a high temporal resolution of two weeks. This design not only expands the incorporation of time complexity into our network, initially introduced by Wang and Ma 2011, but also facilitates the capture of dynamic changes in plant development and physiological characteristics over time. Using data from different growth stages allows us to explore the contribution of each stage to prediction accuracy and gain insight into the relative importance of growth stages in determining plant phenotypic traits.

To our knowledge, the integration of multispectral and thermal data, coupled with an in-depth analysis of the complete growth cycle, remains novel in the context of automated phenotyping for yellow rust score prediction. Our research aims to fill this gap and offer valuable insights into the feasibility, potential benefits, and implications of these approaches.

Despite the ongoing advances in deep learning techniques, basic methodologies continue to provide effective tools for predicting phenotypes. The decision to employ machine learning methods for prediction tasks depends on factors such as technical capabilities, computational resources, and monitoring infrastructures. Notably, basic correlation methods can still yield accurate predictions, as discussed earlier. Our scientific challenge lies in investigating whether a basic approach could suffice to predict the abundance of yellow rust based on a ground-truth target.

We anticipate a significant challenge arising from the relatively restricted dataset, encompassing a total of 1064 expertly assessed scores along with their associated spectral measurements. This limitation poses a hurdle in crafting a well-generalising model. Nevertheless, we intend to surmount this obstacle by employing a combination of anti-overfitting techniques. Our goal is not only to address the immediate challenge but also to propose an approach that can be adopted in subsequent studies. We aim to provide a clear and reusable pipeline for creating a training dataset and effectively training it.

The project encompasses significant challenges in terms of data management and extraction, particularly considering the influence of time and geographical factors. Our approach offers a potential advantage compared to others, as we aim to develop machine learning tools that have the potential to predict disease scores throughout the entire wheat plant growth cycle, without relying solely on commonly used data sets and offering new avenues to scale up phenotyping in large-scale fields.

## 1.5 Structure

The subsequent chapters feature a comprehensive review of the literature, accompanied by a detailed explanation of the tools used in this study. Subsequently, we will thoroughly account for the data acquisition and processing methodologies. Moreover, we will conduct a benchmark comparison between the restricted and unrestricted datasets, followed by a comprehensive evaluation of the machine learning outcomes. Lastly, we will discuss the findings of this study and explore further avenues for research and insight generation.

# 2 Related Work

The following sections provide an overview of the scientific context of this thesis. We will begin by presenting a historical background on remote sensing and its significance in ecological research. Subsequently, we will explore the applications of unmanned aerial vehicles (UAVs) in remote sensing for agricultural purposes, highlighting the comparison between traditional phenotype acquisition methods and remote sensing-based approaches. Finally, we will delve into the emergence of novel machine learning techniques for yellow rust prediction, with a focus on the indispensability of deep learning in UAV-based applications. Emphasis will also be placed on the importance of time series analysis within the machine learning framework.

## 2.1 Satellite Remote Sensing

Remote sensing, the process of scanning the Earth using satellites or high-flying aircraft to gather information, has witnessed significant development since the 1960s. As Slotten reports (Slotten 2002), the space race between the Soviet Union and the United States played a crucial role in advancing satellite technology. Initially driven by the need for intelligence and military surveillance, satellite-based remote sensing technology rapidly evolved, as extensively discussed by Cloud and Clarke (Cloud and Clarke 1999). In 1957, the Soviet Union launched Sputnik 1, the world's first artificial satellite, followed by the successful launch of Explorer 1 by the United States in 1958. These groundbreaking events marked the inception of satellite-based remote sensing and laid the groundwork for subsequent advancements in the field. Following the successful launch of TIROS-1, the inaugural meteorological satellite in 1960, a notable milestone was achieved, allowing for the capture of images that facilitated ecological and geographic research. Subsequently, the United States embarked on the groundbreaking Landsat program in 1972, marking the initiation of the first Earth resource satellite initiative (Wulder et al. 2019). This program was designed to gather data from the Earth using remote sensing techniques. Over the past 45 years, the Landsat program has been ongoing, culminating in the launch of its most recent satellite, Landsat 8, in 2013.

Initially focused on military operations, remote sensing quickly found its way into civil research. In fact, since the Landsat 1 launch in 1972, remote sensing has undergone remarkable advancements. Today, the field has experienced a transformative evolution, with a multitude of satellites orbiting the Earth, equipped with cutting-edge sensors and imaging technologies. These advances have revolutionised remote sensing capabilities, enabling precise and comprehensive data collection and analysis across various applications.

## 2.2 Emergence of UAV Remote Sensing

The reliability of satellite technology has paved the way for the development of useful tools such as the Google Maps satellite view. However, the limitations of this technology become apparent when higher image resolutions are required. While high-resolution satellites like QuickBird, which was operational till 2015, offered resolutions of up to 0.6 meters 1, commonly used satellite networks like the European Sentinel operate within a spatial resolution range of 10 to 60 meters. The WorldView fleet, on the other hand, provides commercially available panchromatic imagery of 0.46 m resolution, and eight-band multispectral imagery with 1.84 m resolution representing one of the highest available space-borne resolutions on the market (E.S.A. 2023), although expensive. Those satellite networks play a crucial role in covering expansive areas surpassing high-resolution satellites' coverage capability. This is primarily due to the operational costs associated with the latter; this trade-off is illustrated by Valenzuela (Valenzuela et al. 2022).

An active line of research in the field is currently trying to further enhance that measures[1], that are not costly or publicly available, using Enhanced Super-Resolution Generative Adversarial Network (ESRGAN). ESRGAN, a deep learning model based on Generative Adversarial Networks (GANs), enhances image resolution by training a generator network to generate high-resolution images from low-resolution inputs, ultimately producing visually appealing results (Salgueiro Romero et al. 2020). Yet again, the results in this field are noteworthy. For example, an image acquired with a resolution of 10 metres is enhanced by a factor of 4 (Soufi and Belouadha 2023). Since the information is synthetically generated, some domain-specific high-resolution operations might still encounter difficulties and challenges with this approach. Even acknowledging those performances, they are not sufficient for several modern-day tasks. In the larger context of ecological research, there are many cases where even higher resolutions are needed. For example, to count canopy heads in a wheat field (see Ma et al. 2022) or to monitor and classify wetland areas (Martins et al. 2020).

In the case of counting canopy heads, the small size and close proximity of the heads make their identification and enumeration a complex undertaking. Similarly, in the context of monitoring wetland areas, the detection and classification of smaller patches require higher-resolution images to ensure comprehensive coverage and accurate identification.

In this context, the use of Unmanned Aerial Vehicle (UAV) scanning has emerged as a pivotal technique for acquiring high-resolution images. This topic has been extensively explored in the past two decades and documented in the comprehensive study conducted by Alvarez (Alvarez-Vanhard et al. 2021). By strategically controlling the flight altitude of the UAV, researchers can capture images at remarkably fine resolutions, enabling them to obtain complex and detailed visual data. Resolution is not

[1]This is an important line of research, and this introduction reports that as insight and not as its goal

the sole advantage derived from the implementation of this technology. There are other significant benefits, such as the ease of acquiring the data without the need for costly bookings of measurements. Additionally, the measurements can be conducted with flexibility, taking advantage of favourable weather conditions.

However, this approach still presents limitations, for instance, stable weather conditions are required to ensure uninterrupted drone flight, as gusts of wind can disrupt operations. Furthermore, for extensive applications, the battery capacity of the drone may present challenges. However, ongoing research is actively addressing this problem through the optimisation of pathing options and battery replacement techniques (Saha et al. 2011). The exponential increase in data volume resulting from finer image resolution poses notable limitations. As captured images become finer in resolution, the volume of data expands exponentially, giving rise to significant challenges in storage, transmission, and computational requirements. Managing and processing such substantial data necessitates careful consideration of storage capacities, data transfer bandwidth, and computational resources. Practical implementations often confine operations within a resolution range of 1 meter to 1 centimetre. This range effectively balances the capture of fine-grained details with the constraints associated with data size and processing capabilities (Aasen et al. 2018).

The integration of UAVs in the field of image acquisition and analysis exhibits significant promise (Nex et al. 2022). Leveraging the potential of UAVs allows researchers to unlock high levels of spatial detail in large-scale applications, providing better images to characterise the phenomena under investigation. Consequently, further exploration of UAV-based acquisition methodologies represents an avenue for advancing the field of precision agriculture and phenotyping.

## 2.3 Precision Agriculture

Precision agriculture, a term that emerged in the 1980s, has since gained significant recognition as a transformative scientific discipline. This multidisciplinary field encompasses the integration of technologies and principles aimed at enhancing crop performance while simultaneously mitigating environmental impact. In practice, precision agriculture research aims to establish a decision support system (DSS) for comprehensive farm management, focusing on optimising input returns while simultaneously safeguarding resources and mitigating pollution (Ceccarelli et al. 2022). One particular area of interest within this field is phenotyping, which has witnessed notable advancements in automation. In traditional agricultural practices, crop state assessment is based primarily on indices and manual measurements. However, the advent of automated airborne phenotyping has introduced a range of benefits over conventional approaches. Foremost among these advantages is its high-throughput capability, enabling rapid data collection across large numbers of plants. By bypassing the phenotyping bottleneck that often hinders breeding programmes, this technology accelerates the pace of genetic gain (Gill et al. 2022). [2]

Following the overview proposed by Xie (Xie and Yang 2020), automated remote phenotyping has several important advantages. A key benefit is its non-destructive nature, which allows repeated measurements on the same plants over time. In contrast, traditional methods can be laborious, time-consuming, and costly and might require the destruction of the plant to analyse it. The use of automated airborne phenotyping also facilitates the acquisition of high-resolution data, which is essential for evaluating and selecting the most promising cultivars in crop breeding and phenotyping and identifying biotic and abiotic stress sources, like pests or water scarcity. Additionally, this technology contributes to improved accuracy by enabling the identification of crucial genes to enhance photosynthesis and reduce overall stress. As the crop phenotype emerges from the intricate interplay between genetic makeup and environmental factors, the high-throughput nature of remote sensing serves as a tool for the timely detection and monitoring of crop responses.

At the beginning of the 20th century, precision agriculture's full potential had yet to be realised due to inadequate consideration of the space-time continuum in crop production, which adds complexity to its implementation (Pierce and Nowak 1999). However, considering advances in both remote sensing technology and automation, significant progress has been made in the past two decades to increase acquisition rates and the resolution of acquired images. Moreover, the new airborne and spaceborne platforms acquire images that are processed by software capable of adapting, rectifying, and radiometrically calibrating raw images, boosting their overall quality. Concurrently, a new set of best practices has emerged (Ozdogan et al. 2010), providing comprehensive and timely coverage of agricultural fields and exploring the value of archived data for temporal image comparisons. Within this context, machine learning has naturally found a role, further enhancing the phenotyping capabilities of precision agriculture tools.

## 2.4 UAV Sensors for Precision Agriculture

UAV-based phenotyping incorporates various sensors, including multispectral, thermal, hyperspectral, and standard RGB. These sensors enable a range of applications in agriculture.

Multispectral data collected by UAVs offer significant advantages, allowing farmers to effectively monitor crop health and stress (Virnodkar et al. 2020), detect diseases and pests promptly (Prabhakar et al. 2011), and optimise crop yields (Zhou et al. 2021). Unlike traditional RGB images, multispectral data capture information across different spectral bands, including green, red, and near-infrared (NIR). This enables the calculation of vegetation indices such as NDVI and GNDVI. Using these indices,

[2]Note that increasing the pace of genetic gain is the ultimate goal of the WheatVIZ project

farmers can identify areas of the field that experience stress or exhibit low vegetation density.

Hyperspectral remote sensing plays a role in solid monitoring, in particular: mineral identification, nutrient, organic carbon, moisture, salinity, and soil texture of agronomic and ecological systems (Yu et al. 2020). Hyperspectral sensors typically operate in the visible and near-infrared (VNIR) wavelength range of 450 nm to 10,000 nm. Nevertheless, some applications cover a wide spectral range, ranging from ultraviolet (0.35 $\mu$m circa) to thermal infrared (12 $\mu$m circa).

Thermal remote sensing, conducted predominantly within the wavelengths of 3-5 $\mu$ m and 8-14 $\mu$ m, has potential, as it is a method to assess leaf pathogens by measuring the temperature of plant leaves(Oerke et al. 2005).

UAVs can be used to generate an orthomosaic (referred to as reflectance maps alternatively), a high-resolution georeferenced image created by combining multiple aerial or satellite images of a particular geographic area.

## 2.5 Machine Learning for Precision Agriculture

This subsection comprises two distinct segments. The first part involves an exploration of common practices and techniques utilised in precision agriculture that leverage machine learning. The second part focuses on a review of specific approaches used in predicting yellow rust.

### 2.5.1 Machine Learning in Phenotyping

Machine learning plays a pivotal role in the agricultural revolution (Tantalaki et al. 2019). It empowers machines to learn and make predictions without explicit programming. Combined with modern sensors and high-tech machinery, machine learning offers diverse applications in agriculture. According to Sharma et al. 2020, these applications include predicting soil parameters such as organic carbon and moisture content and plant parameters, such as crop yield, diseases and water status. While the traditional methods employed by plant breeders can monitor important vegetation parameters, they may suffer from inherent approximations and compromised accuracy. One widely used method is the analysis of the Normalised Difference Vegetation Index (NDVI) (Huang et al. 2021). The NDVI calculates the ratio between the "red" spectral band (600-700 nm) and the "near infrared" spectral band (700 - 1300 nm), leveraging the reflection or absorption characteristics of vegetation in these bands. Through the NDVI, essential parameters such as leaf area index (LAI) or chlorophyll content can be derived through correlations. However, machine learning techniques can be employed for a more precise determination of these parameters. Deep learning methods outperform basic approaches in phenotyping due to their ability to handle vast amounts of noisy and diverse data (Ansarifar et al. 2021).

These techniques provide rapid and precise analysis crucial for high-throughput phenotyping. For example, Li et al. 2019 achieved an r-square[3] score of 0.84 using a random forest predictor for leaf area index estimation from images acquired by unmanned aerial vehicles (UAVs), getting a better measurement than a correlation with the NDVI.

Furthermore, deep learning approaches can generate predictions for intricate and uncertain phenomena, which is essential in plant stress phenotyping from both biotic and abiotic sources. This, along with advancements in computational capabilities, has led to the prominence of deep learning methods. Deep learning, coupled with computer vision, facilitates the classification of crop images, enabling monitoring of crop quality and assessment of yield and water stress (Wang et al. 2023; Chandel et al. 2022). Convolutional neural networks (CNNs) play a crucial role in image-processing machine learning pipelines, particularly in feature extraction. The convolution operation, achieved through the application of filters to small regions of input data, generates feature maps that capture local patterns and structures. This spatial information helps the network learn important features and extract higher-level representations, which is valuable in precision agriculture applications. Ansarifar et al. 2021 proposed a model stating that: "plant phenotype is determined by genotype (G), environment (E), management (M), and their interactions (G × E × M)". Deep learning methods excel in capturing the inherently nonlinear interactions among these variables. Singh et al. 2018 reviewed various deep learning architectures, such as AlexNet, GoogLeNet, VGG CNN, and Inception-v3, that have been successfully employed to identify and classify diseases and stresses in plants. To address the challenge of training large deep models, transfer learning and fine-tuning techniques have been utilised, enabling high accuracy even with limited training datasets. Deep learning models have been successfully applied to detect diseases and pests in different crops, including tomatoes, apples, soybeans, and cassava. For instance, Mohanty et al. 2016 trained a deep CNN using a publicly available dataset of 54,306 images of diseased and healthy plant leaves, achieving an accuracy of 99.35% in identifying 14 different crop species and detecting 26 diseases on a separate test set. Deep learning models have also been developed for quantifying plant stress severity and predicting water stress. Giménez-Gallego et al. 2019 proposed an automatic drought detection system for the middle growth stage of maize, utilising Gabor filters for texture feature extraction and a convolutional neural network for feature extraction and classification. The experimental results showcased an average recognition rate of 98.84%. In the review by Zhou et al. 2021, which focused on modelling crop water stress, basic statistical regression methods were found to have limitations due to the non-linear relationship between the crop water stress index (CWSI) and physiological indicators. They also showed that

[3]The $r^2$ score, or the coefficient of determination measures the proportion of variance in the dependent variable explained by the independent variables in a regression model, ranging from 0 - indicating a poor fit - to 1 - representing a perfect fit).

deep learning models, on the other hand, provide a reliable alternative for capturing complex associations and have been successfully employed to predict the CWSI using various environmental factors. Furthermore, they claim that the integration of multispectral sensors with thermal cameras holds promise for a comprehensive evaluation of crop physiological conditions and stress.

### 2.5.2 Machine Learning for Yellow Rust Prediction

Transitioning from phenotyping in general to solutions tailored for yellow rust predictions, the following section provides an overview of the primary methodologies for yellow rust prediction. The underlying principle is to move beyond predictions limited to individual leaves and embrace broader generalisations across the entire field.

**Single leaf analysis and ground image acquisition**
One of the earliest studies to forecast yellow rust using machine learning was undertaken by Moshou et al. 2004. In this study, high-resolution images of wheat canopies were input into a Self-Organising Map (Kohonen 1990) to map the high-dimensional images onto a one-dimensional discrete lattice of neuron units, extracting the most relevant features. This feature representation was then fed into a multi-layered perceptron (MLP) to predict over two classes. The significance of this paper lies in showcasing the superiority of deep learning over basic machine learning, with the Multi-Layered Perceptron (MLP) outperforming the Bayes selector by 4%, achieving a final accuracy of 99%.

A crucial observation here is the ease of developing a model analysing leaf-level images to predict whether a leaf is infected. This task is inherently straightforward, given the model's ability to recognise disease patterns akin to an expert. Supporting this, numerous studies have demonstrated efficiency in predicting yellow rust on a leaf level, for instance Kukreja and Kumar 2021, which utilised 1486 wheat plant images and 514 wheat stripe rust images acquired by hand-held devices. The study proposes a Deep Convolutional Neural Network (DCNN) for wheat rust disease classification, effectively distinguishing between healthy and diseased plants. The architecture involves convolution, pooling, and fully connected layers, achieving a 97.16% classification accuracy for wheat rust diseases.

Additionally, Koc et al. 2022 illustrated the processing of phenocart-acquired[4] data together with expert-acquired disease scores, in a similar fashion as the scores employed in this thesis. The raw data acquired by the phenocart are 119 spectral image-based predictors. After automatic outlier removal and handling of missing data, the dataset had 439 and 505 instances at two time points. To train the model,

[4]A phenocart is a mobile platform with sensors and imaging devices designed for efficient, non-destructive high-throughput phenotyping in agriculture, providing detailed data on plant traits and health conditions to support research in crop improvement and breeding programs.

recursive feature elimination, employing random forest regression, was used for supervised feature selection. The final models were trained, tuned, and evaluated. The results were that phenotypic characterisation showed considerable yellow rust infection, with correlation analyses revealing associations between sensor data and disease scores. Recursive feature elimination identified important predictors, predominantly the commonly used spectral indices (e.g.; NDVI). The Random Forest models achieved prediction accuracies of 0.50 and 0.61 for the datasets associated with both time points, demonstrating a linear trend between observed and predicted scores. In the broader frame of the research, this study hinders the direction that the different time steps might contribute differently to the prediction of yellow rust. One of the objectives of this thesis is to verify whether this can be exploited for better predictions.

Moreover, Mi et al. 2020 introduces a comprehensive approach to wheat stripe rust disease grading, assigning grades from 0 (healthy) to 5 (severely diseased). It utilises deep learning, incorporating a C-DenseNet network architecture enriched with a Convolutional Block Attention Module (CBAM). Image preprocessing and augmentation techniques, including a blade mask method for cropping and data augmentation, contribute to improved model performance. The proposed architecture outperforms classical ResNet, showcasing accuracy (24%), precision (25%), recall (25%), and F1 score (26%). The study also underscores the effectiveness of placing attention modules between dense blocks.

These successful studies demonstrate the feasibility of predicting the degree of yellow rust infection using plant images at the leaf level. However, a scientific challenge arises when attempting to scale up to predict infection across a set of plants in a field trial.

**UAV-based acquisition**

Two studies that employ Unmanned Aerial Vehicle (UAV)-based images are presented. In Tang et al. 2023, a UAV flies at a height of 1.25 meters above wheat canopies, acquiring high-resolution field images. The study employs a RustNet model based on ResNet-18 for predicting wheat stripe rust through image classification. The ResNet-18 architecture, pre-trained with ImageNet data, is fine-tuned for two classes, effectively distinguishing disease and non-disease classes. Training involves tile images resised to $224 \times 224$ pixels, with parameters retrained over 100 epochs using the Adam optimiser. The architecture incorporates shortcut connections in residual blocks, with a Grad-CAM method for visualisation, providing insights into critical regions for disease prediction. Image labelling, facilitated by the Rooster software, employs a semi-automatic approach, enhancing efficiency and accuracy through cyclical supervision. RustNet demonstrates robust performance across diverse scenarios, including independent validation in various locations, platforms, and wheat types. The study, involving manual labelling of 56 images, reveals that RustNet's accuracy for stripe rust detection progressively increases through semi-automatic image labelling, reaching an Area Under the Curve[5] (AUC) of 0.85 in Stage 3. Comparative performance analysis with 20,360 tile images showcases AUC values of 0.64, 0.78, and 0.87 for RustNet at Stages 1, 2, and 3, respectively, demonstrating the effectiveness of automatic labels and the potential for consistent results through thorough ResNet18 training during RustNet development.

In contrast, Zhang et al. 2019 employ hyperspectral data collected at a mid-altitude flight altitude of 30 meters to forecast the presence of yellow rust. This prediction occurs at an individual pixel level, utilising a ResNet neural network and automatically labelled data as targets. Pixels are categorised based on the NDVI assessment, initially distinguishing between ground and vegetation before grouping pixels into rust-affected areas and healthy regions during the yellow rust occurrence. Pixels with an NDVI value exceeding 0.3 are labelled as rust or healthy, while the rest are categorised differently. The study utilises 10,000 hyperspectral image blocks for training and validation (80%) and 5,000 blocks for testing the proposed model's performance. Results reveal that a model with four Inception-ResNet blocks outperforms other configurations. In comparison with a spectral-based basic machine learning method (random forest), the proposed Deep Convolutional Neural Network (DCNN) achieves higher accuracy (0.85 vs. 0.77). Additionally, the DCNN demonstrates improved accuracy for yellow rust detection, particularly in the later stages of the crop growth season, reaching a recall rate of 0.86 for the rust class on datasets collected on 15 May 2018.

Similarly, very recently (mid-way through the thesis, since we started in March), Nguyen et al. 2023 adopted a similar set-up as the one presented in this study for the spring wheat. The startling difference is the fact that the flights were conducted at an operative height of 20 m, therefore our interest in seeing what was employed to identify healthy, mildly infected, and severely infected wheat plots. Additionally, a custom 3-dimensional convolutional neural network (3D-CNN, consisting of 3 convolutions intervalled by max-pooling layers and an MLP on top of this feature map) achieved a 60% detection accuracy as early as 40 days after sowing, increasing to 79% for the spectral-spatio-temporal fused data model. This model aggregated the image dataset with images acquired at different times, to train the deep learning model. Our area of interest is explicitly including the temporal information within the model, enabling even higher pattern recognition

In a parallel development, emerging midway through our thesis that commenced in March, Nguyen et al. 2023 embraced a setup resembling the one outlined in our study for spring wheat. The key deviation lies in the operational altitude of 20 meters during their flights, prompting our exploration into the viability of utilising a flight height of 60 meters for identifying infected wheat plots. Additionally, Nguyen et al.

[5]The AUC is a metric used in ML to assess the performance of a binary classification model through the Receiver Operating Characteristic (ROC) curve. A higher AUC, ranging from 0 to 1, signifies better discrimination, with 0.5 indicating performance equivalent to random chance and 1.0 representing perfect discrimination.

categorised wheat into three classes—healthy, mildly infected, and highly infected; while our objective is to establish a regressor spanning a score range of 1 to 9, allowing for a more nuanced disease descriptor.

Concerning the model, a custom 3-dimensional convolutional neural network (3D-CNN), consisting of three convolutions interspersed with max-pooling layers and a multilayer perceptron (MLP) atop the resultant feature map, achieved a 60% detection accuracy as early as 40 days after sowing. This accuracy rose to 79% for the spectral-spatio-temporal fused data model, integrating the image dataset with images acquired at different times to train the deep learning model. Significantly, our emphasis explicitly extends to the incorporation of temporal information within the model, facilitating heightened levels of pattern recognition.

To conclude this section it is important to mention Wang and Ma 2011. Their study focuses on the temporal dimension of the yellow rust regression problem. In fact, instead of utilising spectral data, this study relied on weather data and historical readings of yellow rust abundance in trial fields in China. By employing kernelised SVM, the study achieved high accuracy, demonstrating the method's effectiveness in predicting the disease based on past-year data. Building on this success and capitalising on the subsequent advancements in drone technology. We think that leveraging multi and hyperspectral data for monitoring yellow rust levels within the yearly growing season might be beneficial to innovate this field.

# 3 Background

The scientific background of this study encompasses a diverse array of concepts central to the investigation of wheat spectral measurement and predictive modelling. As we delve into the intricacies of agricultural research, it is important to establish a foundational understanding of key components that underpin our exploration. This section provides a broad overview of the scientific landscape, encompassing spectral measurement, data preprocessing techniques, predictive models, and evaluation metrics.

## 3.1 Spectral Measurement and NDVI

Here we briefly report some of the most common spectral bands acquired in precision agriculture (Lu et al. 2020). We employed those to create the dataset for the disease score prediction.

Table 1: Spectral bands acquired in the Wheatviz project.

| **Channel Description** | **Wavelength** | **Use in Precision Agriculture** |
|---|---|---|
| Red (R) | 620-750 nm | Monitoring plant health, identifying stressed vegetation. |
| Green (G) | 495-570 nm | Assessing chlorophyll content and plant vigor. |
| Blue (B) | 450-495 nm | Analysing water content in vegetation. |
| Red Edge (RE) | 690-750 nm | Detecting subtle changes in plant health and stress. |
| Near Infrared (NIR) | 780 - 2500 nm | Provide high-resolution imagery for detailed field analysis. |
| LWIR (Long-Wave Infrared) | 8-14 µm | Measuring temperature variations in crops and soil. |

Those spectral measurements already carry useful information concerning the state of plants. Nevertheless, we also computed the NDVI since it is the most commonly used vegetation index in remote sensing. It quantifies the amount of live green vegetation in an area based on the reflectance of light in the visible and near-infrared spectral regions. The NDVI is also specifically instrumental in detecting changes in plant growth over time. When analysed through time, NDVI can reveal where vegetation is thriving and where it is under stress (Berra et al. 2019).

The formula for calculating NDVI is as follows:

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

where:

- NIR: represents the reflectance in the near-infrared band.
- Red: represents the reflectance in the red band.

NDVI values range from -1 to 1, with higher values indicating healthier and more abundant vegetation. Negative values often represent non-vegetated surfaces like water or barren land, while values near zero may indicate sparse or stressed vegetation. Positive values close to 1 suggest dense and healthy vegetation.

## 3.2 Data Preprocessing

In this subsection, we introduce techniques for transforming quadrilateral images into rectangular forms, emphasising the role of kernel functions and their impact on the reconstructed image. This part in instrumenthal to the dataset creation from the multispectral orthomosaics.

### 3.2.1 Projective Transformation

This thesis utilises a projective transformation to reshape an image from a quadrilateral to a rectangular form, a frequent task in image processing and computer vision. The projective transformation, distinguished by its non-linear mapping of points from one perspective to another, holds substantial value in machine vision by fostering a more regular input and substantially enhancing generalisation (Fan et al. 2022). Furthermore, this method confers a notable advantage by facilitating the creation of a standardised dataset comprising annotated images, a valuable resource for future research endeavours. The spatial operation can be delineated into two distinct phases: homography and resampling.

**Homography**
Mathematically, a homography is described as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

where $(x, y)$ are the original coordinates in the source image, $(x', y')$ are the transformed coordinates in the destination image, and $M$ is the 3x3 transformation matrix:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

To find the values of the transformation matrix $M$ in the projective transformation equation 1, you typically use a set of corresponding points in both the source and destination images. Easy examples are the extremal points of two quadrilaterals. It follows a set of linear equations that solves for the elements of $M$.

Let's consider a set of $n$ corresponding points:

$$\begin{aligned} (x_1, y_1) &\mapsto (x'_1, y'_1) \\ (x_2, y_2) &\mapsto (x'_2, y'_2) \\ &\vdots \\ (x_n, y_n) &\mapsto (x'_n, y'_n) \end{aligned}$$

For each corresponding pair, two equations (one for $x$ and one for $y$) can be set up:

$$\begin{aligned} x'_i &= m_{11}x_i + m_{12}y_i + m_{13} \\ y'_i &= m_{21}x_i + m_{22}y_i + m_{23} \end{aligned}$$

This can be written in matrix form as:

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

For all $n$ corresponding points, you can stack these equations into a single matrix equation:

$$\begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ \vdots & \vdots \\ x'_n & y'_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

This system of equations can be solved using various techniques such as the Singular Value Decomposition (SVD) method. The solution will provide the values of the elements in the matrix $M$. In Python, libraries like OpenCV 2 often provide functions to find the homography matrix, which encapsulates this process and can handle the solution numerically. In the provided code, "`cv2.findHomography`" is used to compute the homography matrix given the corresponding points.

**Resampling**
The previously discussed section addresses the recalibration of each pixel's position in the new plane. However, adjusting pixel locations falls short of achieving a comprehensive image transformation. To complete this process, the intensity of each pixel in the transformed image ought to be computed, necessitating the application of resampling and reconstruction techniques.

Resampling plays a critical role in determining the intensity values of pixels at non-integer coordinates within the transformed image. This becomes particularly crucial since, in general, the transformed coordinates $(x', y')$ are not integers, whereas pixel values are defined at discrete integer coordinates. Resampling methods estimate pixel intensities at these non-integer coordinates by leveraging information from nearby integer coordinates. The determination of pixel weights in this context is facilitated by kernel functions (refer to Section 3.2.2). These kernel functions inherently possess finite support, indicating their nonzero values exclusively within a specific neighbourhood of the point being interpolated. The calculated weighted sum of pixel intensities subsequently furnishes the estimated intensity at the non-integer coordinate.

In mathematical terms, the resampling operation can be expressed as follows:

$$I'(x', y') = \sum_{x,y} I(x, y) \cdot K(x' - x, y' - y)$$

where,

- $I'(x', y')$ denotes the value of the resampled image at the new coordinate $(x', y')$.
- $I(x, y)$ represents the original pixel value at the coordinates $(x, y)$ in the source image.
- $K(x' - x, y' - y)$ refers to the resampling filter or the kernel function. This function delineates how values from the original image are weighted and amalgamated to compute the new value at $(x', y')$, contingent upon the specific resampling method utilised.

This formulation affords a discernible advantage: each reconstructed pixel encapsulates information from proximate pixels, ensuring a seamless transition between the quadrilateral image and its rectangular projection. The following section will illustrate the four kernel functions employed in this thesis.

### 3.2.2 Resampling Kernel Functions

Kernel filters dictate how original image pixels contribute to generating new pixel values. Here, we introduce the four main kernels utilised:

**Nearest neighbours**: this method, while computationally efficient, assigns the value of the nearest source pixel to the destination pixel. It is commonly chosen for its speed, yet it may introduce blocky artifacts, especially when upscaling images (Studley and Weber 2011). The operation is represented as follows:

$$I'(x', y') = I(\text{round}(x'), \text{round}(y'))$$

where $I'(x', y')$ denotes the value of the resampled image at the new coordinates $(x', y')$ and $I(x, y)$ represents the original pixel value at coordinates $(x, y)$ in the source image.

**Bilinear**: bilinear interpolation computes the new pixel value as a weighted average of the four nearest neighbouring pixels, offering smoother transitions compared to nearest neighbours. It strikes a balance between simplicity and image quality, making it suitable for real-time applications or web graphics:

$$I'(x', y') = (1-\alpha)(1-\beta)I(x, y)+\alpha(1-\beta)I(x+1, y)+(1-\alpha)\beta I(x, y+1)+\alpha\beta I(x+1, y+1)$$

where $\alpha$ and $\beta$ are the interpolation weights of $x'$ and $y'$ respectively.

**Bicubic**: bicubic interpolation employs a more sophisticated model, calculating new pixel values based on a weighted average of surrounding pixels. It uses a larger context window than the bilinear kernel. This results in smoother transitions and higher-quality resampled images, making it ideal for applications where image fidelity is crucial, such as photography and graphic design:

$$I'(x', y') = \sum_{i=-1}^{2} \sum_{j=-1}^{2} I(x + i, y + j) \cdot w(i, x' - x) \cdot w(j, y' - y)$$

where $w(i, x'-x)$ and $w(j, y'-y)$ are the interpolation weights, calculated based on the distance between the new coordinates $(x', y')$ and the original pixel coordinates $(x, y)$.

**Lanczos**: Lanczos interpolation is renowned for its ability to preserve image details by employing a weighted sinc function. It ensures accurate resampling while minimising aliasing artifacts, making it suitable for resising images with fine details and sharp edges. However, it may demand more computational resources compared to simpler methods:

$$I'(x', y') = \frac{\sum_{i=-a}^{a} \sum_{j=-a}^{a} I(x+i, y+j) \cdot \text{sinc}(x' - x - i) \cdot \text{sinc}(y' - y - j)}{\sum_{i=-a}^{a} \sum_{j=-a}^{a} \text{sinc}(x' - x - i) \cdot \text{sinc}(y' - y - j)}$$

where $a$ represents the width of the chosen window function. Moreover, sinc$(x' - x - i)$ and sinc$(y' - y - j)$ are the sinc functions, which interpolate the pixel values based on their distances from the original and resampled coordinates.

As a summary, Table 2 presents how many pixels each interpolation method uses in reconstructing the signal:

Table 2: Summary of interpolation methods and neighbouring pixels used.

| **Interpolation Method** | **Number of Neighboring Pixels Used** |
|---|---|
| Nearest neighbour | Only the neighbouring pixel at the closest integer coordinates. |
| Bilinear | 4 neighbouring pixels at the corners of the nearest integer coordinates. |
| Bicubic | 16 neighbouring pixels within a 4x4 grid around the nearest integer coordinates. |
| Lanczos | Variable number of neighbouring pixels determined by the chosen Lanczos window size (a), typically more than the bicubic approach. |

### 3.2.3 Feature Scaling

Feature scaling, also known as min-max normalisation, is a key technique in machine learning.

Let $I$ be the original pixel intensity in an image with values in the range $[I_{\text{min}}, I_{\text{max}}]$. The min-max normalisation for image pixel values is given by:

$$I' = \frac{I - I_{\text{min}}}{I_{\text{max}} - I_{\text{min}}}$$

where:

- $I'$ is the normalised pixel intensity.
- $I_{\text{min}}$ is the minimum pixel intensity among all in the original image.
- $I_{\text{max}}$ is the maximum pixel intensity among all in the original image.

This normalisation process ensures that pixel intensities are re-scaled to a range between 0 and 1, facilitating consistent representation across different images. Ensuring uniform scales across images is a fundamental preprocessing step critical to the stability of the optimisation process in our model. The occurrence of large plateaus[6] within the loss function landscape can pose significant challenges during training since they can impede the effective updating of model parameters. This phenomenon is especially problematic when the norms of the gradients computed during back-propagation approach zero, leading to what is commonly referred to as the "vanishing gradient" problem. When gradients vanish, the model struggles to learn and adapt to the nuances present in the data, hindering the overall training efficacy.

By maintaining uniform scales across images through min-max normalisation (as detailed in Section 3.2.3), we mitigate the risk of encountering these plateaus. This normalisation process ensures that all input features share a consistent range, preventing certain features from dominating the optimisation process due to their scale.

## 3.3 Loss Functions and Metrics

The selection of an appropriate loss function plays a critical role in the training and evaluation of machine learning and deep learning models.

### 3.3.1 Mean Squared Error

In this thesis, the Mean Squared Error (MSE) emerged as the default choice, unless explicitly mentioned otherwise. The reason for this selection lies in the unique characteristics of MSE and its compatibility with machine learning approaches. MSE, as a loss function, quantifies the average squared difference between the predicted and actual values, thereby providing a measure of how well the model's predictions align with the true data. The mathematical representation of MSE is as follows:

$$J_0 = MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where $N$ represents the total number of data points. $y_i$ denotes the actual or ground truth value. $\hat{y}_i$ represents the model's predicted value and $J_0$ is how the loss function will be addressed in the next sections. It is worth mentioning that MSE holds particular significance in deep learning because MSE's mathematical simplicity makes it computationally efficient and easy to optimise during the training process. This is especially important given the computational load of deep network approaches.

[6]Plateaus are regions where the loss function exhibits minimal changes.

Nonetheless, it's important to note that in certain situations, alternative loss functions (e.g., L1 loss) have been considered, but we decided not to use them in our operations.

### 3.3.2 Mean Absolute Deviation

Mean Absolute Deviation (MAD) provides an alternative perspective on prediction accuracy when compared to the Mean Squared Error (MSE). Unlike MSE, MAD measures the average absolute difference between the predictions of our model and the true values. The mathematical representation of MAD is as follows:

$$MAD = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

here, $N$ represents the total number of data points, $y_i$ denotes the actual or ground truth value, and $\hat{y}_i$ represents the model's predicted value.

For both MSE and MAD, lower values signify more accurate predictions. MAD's simplicity in interpretation makes it an insightful metric, offering a clear understanding of the average absolute discrepancies between predicted and actual values.

### 3.3.3 $r^2$ Score

The $r^2$ score, commonly known as the coefficient of determination, plays a crucial role in regression analysis, serving as a key metric to evaluate model performance. Ranging from 0 to 1, the $r^2$ score indicates the proportion of variance in the dependent variable that can be predicted by the model's independent variables. A perfect fit is denoted by an $r^2$ score of 1, implying precise predictions, while a score of 0 suggests the model's inability to explain any variance around the mean.

Mathematically, the $r^2$ score is expressed as:

$$r^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

here, $N$ represents the total number of data points, $y_i$ denotes actual values, $\hat{y}_i$ signifies the model's predictions, and $\bar{y}$ is the mean of the actual values. Lower $r^2$ values indicate less accurate predictions.

### 3.3.4 $r^2$ Score Adjusted for Model Complexity

For deep learning models, we introduce the adjusted $r^2$ score to address concerns related to model complexity. The adjusted $r^2$ score enhances the regular $r^2$ by considering the impact of the number of trainable parameters ($k$) on model performance when training with N data points. This adjustment is particularly relevant in the context of deep learning models, which often involve a higher number of predictors.

The formula for the adjusted $r^2$ score is:

$$r^2_{\text{Adj}} = 1 - \frac{(1 - r^2) \times (N - 1)}{N - k - 1}$$

where $r^2_{\text{Adj}}$ represents the adjusted $r^2$ score, which is a modified version of the coefficient of determination ($r^2$). $r^2$ is the coefficient of determination, representing the proportion of the variance in the dependent variable that is predictable from the independent variables in a regression model. $N$ denotes the number of observations in the dataset. $k$ represents the number of independent variables in the regression model.

Adjusting for the number of predictors penalises the inclusion of excessive or irrelevant predictors, providing a more conservative and balanced evaluation. This consideration becomes especially significant in deep learning, where models tend to exhibit higher complexity and a larger number of predictors. The adjusted $r^2$ enables a more accurate assessment, crucial for discerning the genuine explanatory power of the model.

## 3.4 Models

In this thesis, we present different models and techniques for supervised machine learning. We differentiate between basic machine learning approaches (such as linear models and SVMs) and deep learning approaches. Basic models offer the advantage of quick training, allowing the application of comprehensive hyperparameter selection techniques like grid search. In grid search, coupled with cross-validation, the dataset is divided into multiple folds. Iteratively, one fold is designated as the validation set while the model is trained on the remaining folds. This process is repeated for each combination of hyperparameters, ensuring a robust training procedure. The scores for each opted-out fold are averaged, representing the performance of each hyperparameter configuration. The best configuration is then selected and tested on the remaining test set to provide an unbiased evaluation of the model's generalisation to unseen data.

Linear Regression, a fundamental technique in statistical modelling and machine learning, aims to establish a linear relationship between the independent variable(s) and the dependent variable. It serves as a versatile tool for predicting numerical outcomes based on input features.

Consider a dataset with $n$ observations, where $y_i$ represents the true value for the $i$-th instance of the dataset targets:

$$\text{Minimize: } J_0(\boldsymbol{w}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^T X_i)^2$$

where $J_0$ symbolizes the cost function or objective function to be minimised. $\boldsymbol{w}$ is the vector to be determined and $X$ is the independent variables matrix.

The aim is to determine the values of $\boldsymbol{w}$ that minimize the sum of squared differences across all observations. While this model is crucial, it has its limitations. To address this, two variations are introduced to enhance the efficiency of solving the regression problem.

### 3.4.1 Lasso Regressor

Lasso, which stands for Least absolute shrinkage and selection operator, is a basic regularised version of the much simpler linear regressor. It introduces a regularisation mechanism, encouraging sparsity in the weight coefficients depending on a regularisation constant.

For the lasso regressor, we can formulate the optimisation problem as:

$$\text{Minimise: } J_0(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|^1$$

where $J_0$ represents the chosen original loss function, $\lambda$ is the regularisation constant, $\|\boldsymbol{w}\|^1$ is the L1 loss of $\boldsymbol{w}$ and m is the number of components of the weight vector $\mathbf{w}$.

The goal of this technique is to balance the fit of the model to the data and the sparsity of the coefficients (L1 regularisation). This encourages a parsimonious model with only the most relevant features while reducing overfitting. The regularisation hyperparameter ($\lambda$) controls the trade-off between these two objectives. This regularisation approach is really versatile, and, under different conditions, can be exploited in other contexts, as we will explain later.

### 3.4.2 Ridge Regressor

For the Ridge regressor, the optimisation problem can be formulated as:

$$\text{Minimise: } J_0(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|^2$$

where $J_0$ represents the chosen original loss function, $\lambda$ is the regularisation constant and $\|\boldsymbol{w}\|^2$ is the L1 loss of $\boldsymbol{w}$.

The goal of this technique is to balance the fit of the model to the data and the shrinkage of the coefficients (L2 regularisation). This encourages a model that retains all features but with smaller, less volatile coefficients, reducing the risk of overfitting.

The regularisation hyperparameter $\lambda$ controls the trade-off between these two objectives.

This procedure helps mitigate overfitting effects and might be employed to gain further insights into the model's behaviour. In particular, L1 regularisation has the potential to result in interpretable artificial intelligence tools, as explained in the next section.

### 3.4.3 Support Vector Machine

A Support Vector Machine (SVM) is a robust regression model, that identifying support vectors and constructs a hyperplane to optimise the margin between these support vectors and data points. In the context of regression, SVM aims to minimize the following regularised formula:

$$\text{Minimize: } \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \max\left(0, |y_i - (\boldsymbol{w}^T X_i + b)| - \varepsilon\right) \tag{2}$$

where $\boldsymbol{w}$ represents the weight vector to be optimised, $C$ is a regularisation hyperparameter controlling the trade-off between maximising the margin and minimising the error, $X_i$ denotes the i-th row of the input data, $y_i$ represents the i-th target values vector $\boldsymbol{y}$, n in the total number of instances in the dataset, $b$ is the bias term, and $\varepsilon$ is a positive constant that defines the acceptable margin of error for prediction. Briefly, choosing higher values of $C$ encourages a smaller-margin hyperplane, resulting in better classification of all training points. On the contrary, smaller values of $C$ encourage larger-margin hyperplanes, leading to a more robust model, at the cost of increasing the training loss.

In non-linear regression tasks, SVM is valuable when kernels are employed to bolster the model's generalisation capabilities. The theoretical basis of this approach comes from functional analysis, in particular from *Mercer's theorem* and the *Kernel Trick*. See Appendix F for a formal formulation of these theorems.

Many kernel functions are used, but one of the most common is the Radial Basis Function (RBF), also known as the Gaussian kernel. The formula for the RBF kernel is expressed as follows:

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x'}\|^2}{2\sigma^2}\right)$$

where $K(\boldsymbol{x}, \boldsymbol{x'})$ represents the kernel function's output. $\boldsymbol{x}$ and $\boldsymbol{x'}$ denote the input feature vectors. $\sigma$ is a hyperparameter that controls the shape of the kernel and influences the smoothness of the decision boundary.

The RBF kernel is particularly effective when dealing with non-linear regression tasks as it allows SVM to transform the input space into a higher-dimensional space, making it capable of capturing intricate patterns and relationships in the data.

### 3.4.4 Random Forest Regressor

A Random Forest (RF) Regressor stands as a robust ensemble machine learning model known for its ability to handle intricate non-linear relationships, making it particularly effective for regression problems. In the context of this thesis, the RF Regressor plays a crucial role in addressing the inherent non-linearity embedded in the tabular dataset (see 4.2.3).

The Random Forest model is constructed from multiple individual decision trees, each tailored to optimise predictions for continuous target variables. These decision trees partition the dataset into subsets based on feature values, allowing them to capture complex data relationships. During model construction, a predefined number of decision trees are generated, each utilising a random subset of the training data through bootstrapping and a random subset of input features. As the model is trained, these trees evolve by iteratively dividing the dataset into subsets to make predictions.

The individual optimisation problem for a single decision tree in a Random Forest Regressor revolves around finding the optimal splits at each node to minimize the variance of the target variable. Unlike the traditional classification tasks, the focus of this model is on reducing the variability in predictions rather than minimising impurity[7] (Louppe 2014). The mean squared error (MSE) serves as a common metric to quantify this variance reduction.

Here's a simplified explanation of the individual optimisation problem for each single decision tree:

1. **Objective function:** the optimisation process aims to minimize the variance of the target variable at each node.

$$\text{MSE}(t) = \frac{1}{N_t} \sum_{i \in I_t} (y_i - \bar{y}_t)^2$$

   where $N_t$ is the number of samples at node $t$, $I_t$ is the set of indices of samples in node $t$, $y_i$ is the target value for sample $i$, and $\bar{y}_t$ is the mean target value for node $t$.

2. **Feature Selection:** the algorithm searches overall features and possible split points to find the feature and value that result in the greatest reduction in variance. For each feature, the algorithm considers different split points and calculates the variance reduction for each. The feature and split point that maximise the reduction are chosen.

[7] Gini impurity, used in decision trees for classification, measures the probability of misclassifying a randomly chosen element and is calculated as $1 - \sum_{i=1}^{C} (p_i)^2$, where $C$ is the number of classes and $p_i$ is the proportion of instances in class $i$.

3. **Recursive Splitting:** once the optimal split is found, the dataset is divided into two subsets based on the selected feature and split point. The process is then recursively applied to each subset until a stopping criterion is met, for example reaching a maximum depth of the decision tree.

By iteratively optimising splits at each node based on variance reduction, the decision tree aims to create a structure that effectively captures patterns in the data. This tree-based approach suffers notably from overfitting. Nevertheless, harnessing the potential of the ensemble model, random forest mitigate this overfitting if properly trained. In fact, the final prediction generated by the Random Forest results from aggregating the predictions of all constituent trees: for each input $x$, the ensemble prediction $\hat{y}_{\text{RF}}(x)$ is computed by averaging the predictions of all decision trees in the Random Forest. Mathematically, it is expressed as:

$$\hat{y}_{\text{RF}}(x) = \frac{1}{N_{\text{trees}}} \sum_{i=1}^{N_{\text{trees}}} \hat{y}_i(x)$$

where $N_{\text{trees}}$ is the total number of decision trees, and $\hat{y}_i(x)$ represents the prediction of the $i$-th tree for input $x$. This ensemble approach enhances predictive accuracy and mitigates overfitting, offering a powerful tool for regression tasks.

The ensemble nature of Random Forest enhances its predictive accuracy and generalisation capabilities.

### 3.4.5 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a pragmatic choice for analysing 8-channel multispectral images and are first popularised in the paper "Gradient-based learning applied to document recognition" (LeCun et al. 1998). They excel in dealing with images because they efficiently capture spatial hierarchies across the various electromagnetic measures present in these images. This spatial invariance allows CNNs to detect patterns and structures across channels, regardless of their positions. Operations that extract features from the images can be mathematically expressed as 2D convolutions.

$$(I * K)(x, y) = \sum_i \sum_j I(x + i, y + j) \cdot K(i, j)$$

where $I$ is the input image and $K$ is the convolutional kernel. $I(x, y)$ is the pixel value at position $(x, y)$ in the input image. $K(i, j)$ is the value of the convolutional kernel at position $(i, j)$. $i$ and $j$ range over the dimensions of the kernel. $x$ and $y$ represent the position in the output feature map where the result of the convolution is calculated.

The resulting feature map dimensions, after the input tensor passes through M convolutional layers, represented by width ($W'$) and height ($H'$), are influenced by the convolutional kernel size, stride, and padding applied in each layer. Specifically, using square kernels of size ($K \times K$), with a stride of $S$ and padding $P$ in each layer, the new parameters after traversing a CNN layer are calculated as:

$$W' = \frac{W - K + 2P}{S} + 1$$

$$H' = \frac{H - K + 2P}{S} + 1$$

This iterative process is repeated $M$ times consecutively, determining the dimensionality of the final feature maps. The parameter $M$ is crucial as it directly impacts the granularity of details discernible by the model. In contrast, the number of channels in the output feature map is contingent upon the quantity of filters or kernels used in each convolutional layer. Specifically, with $N$ filters in a layer, the resulting feature map will possess $N$ output channels. The intermediate product within the feature extractor will exhibit the shape [N, W', H'].

In conclusion, CNNs are well-suited for multispectral data as they automatically learn features, obviating the need for extensive manual feature engineering. They also provide regularisation in the form of inductive biases, which is crucial for managing the complexity of such data.

### 3.4.6 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a powerful choice for analysing sequential data, allowing for the modelling of dependencies and patterns over time. These networks efficiently capture temporal relationships and patterns, fully harnessing the information brought by the time series of multispectral images used in this study. It is first introduced in the paper "Long short-term memory" (Hochreiter and Schmidhuber 1997).

The time modelling capabilities of this network are their ability to maintain and update internal states, allowing them to remember and utilise information from previous time steps while selectively forgetting irrelevant details. The core component is the LSTM cell because it manages the flow of information by using a system of gates that control the interactions and updates of the cell state and hidden state. These gates, namely the input gate ($i_t$), forget gate ($f_t$), and output gate ($o_t$), regulate the feature map output by the cell. The functioning of each gate is as follows:

- **Input Gate ($i_t$):** the input gate, operating on both the current input ($x_t$) and the previous hidden state ($h_{t-1}$), determines the inclusion of new information into the cell state ($C_t$). Consequently, for each element in the cell state, the input gate controls the degree to which new information ($\tilde{C}_t$) is assimilated into the cell state ($C_t$). In this component a sigmoid activation ($\sigma$) is used to bound values within the 0 to 1 range.

- **Forget Gate ($f_t$):** the forget gate is responsible for deciding whether to retain or discard information from the previous cell state ($C_{t-1}$), considering both the current input ($x_t$) and the previous hidden state ($h_{t-1}$). This is achieved through element-wise multiplication of $f_t$ and $C_{t-1}$, representing the selective inclusion or exclusion of information from the cell state. Similarly to the input gate, the forget gate employs sigmoid activation to control the extent of information retention.
- **Output Gate ($o_t$):** the output gate, which considers both the current input ($x_t$) and the cell state ($C_t$), feeds information to the upcoming hidden state ($h_t$). Within the output gate, a sigmoid activation regulates the influence of information from the cell state on the hidden state. In addition, a hyperbolic tangent ($\tanh$) activation re-scales the values of the cell state, confining them to a range between -1 and 1 to regulate the output. The final outcome, the new hidden state ($h_t$), is achieved through the element-wise multiplication of $o_t$ and $\tanh(C_t)$.

Mathematically, the above operations within an LSTM cell can be described as follows:

- $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ (Input Gate)
- $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ (Forget Gate)
- $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ (Output Gate)
- $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$ (Cell State Update)
- $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$ (Cell State)
- $h_t = o_t \cdot \tanh(C_t)$ (Output)

where $i_t$, $f_t$, and $o_t$ are the input, forget, and output gates, respectively. $\tilde{C}_t$ represents the candidate cell state. $C_t$ denotes the cell state. $h_t$ is the output of the LSTM cell. $W_i$, $W_f$, $W_o$, and $W_C$ are weight matrices. $b_i$, $b_f$, $b_o$, and $b_C$ are bias vectors.

This mechanism empowers LSTMs to capture and retain long-term dependencies and relationships within sequential data.

### 3.4.7 Residual Network

Residual Networks (ResNet) were initially introduced in the paper “Deep Residual Learning for Image Recognition” (He et al. 2015). The key innovation lies in the introduction of the paradigm of residuals to address the degradation problem encountered in deep networks. The fundamental concept of residual learning is the residual unit. It allows layers to explicitly approximate a residual function, denoted as $F(x) := H(x) - x$, instead of directly fitting the desired mapping $H(x)$. In practice, the network learns to optimise the easier task of adjusting the input $x$ by the residual mapping ($F(x)$) to approximate the desired output as $F(x) + x$.

Figure 2 presents a visual representation of a residual module.

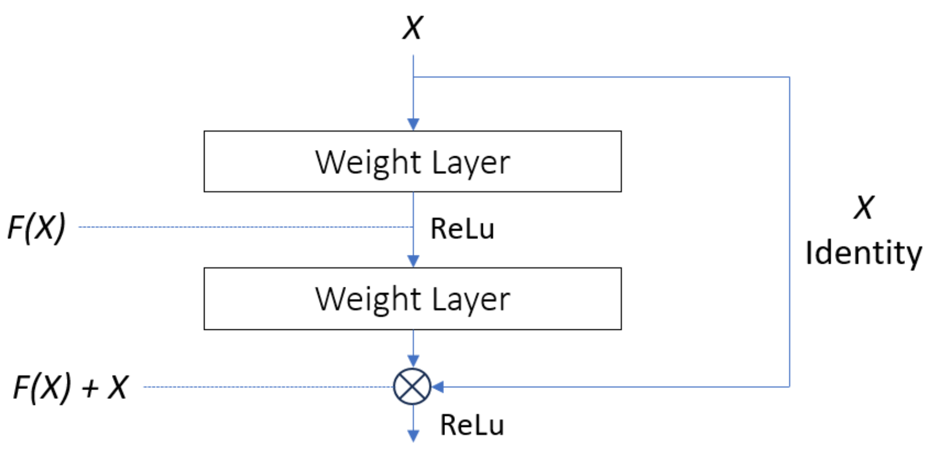


Figure 2: Residual learning block as presented in He et al. 2015. F(X) refers to the passage of X through the first convolutional layer.

Each Residual Module learns a residual function, denoted as $F(x)$, rather than a direct mapping. This approach facilitates the model to learn not just a standard transformation of the feature representation, but also the identity function, $H(x) = x$. Achieving this involves driving $F(x)$ towards zero, which can be accomplished by setting the weights and biases in $F(x)$ to zero. Additionally, ResNets utilise shortcut connections, which enable the gradient to be directly backpropagated to earlier layers, bypassing intermediate layers. This mechanism addresses the vanishing/exploding gradient problem and allows layers that do not enhance performance to effectively approximate an identity function, thus being disregarded during training. These characteristics not only collectively facilitate the successful training of very deep networks but also incorporate identity mappings through shortcut connections, introducing neither extra parameters nor computational complexity.

For our specific implementation, we decided to use ResNet-34 as implemented in Pytorch. The rationale behind it is that it is still a relatively small model, compared to later versions and, at the same time, it outperforms the first version of ResNet, as proposed in He et al. 2015. ResNet-34 is a deep convolutional neural network consisting of 34 layers. Its architecture is outlined as follows:

1. The network begins with a convolutional layer comprising 64 filters, each with a kernel size of $7 \times 7$, followed by a max-pooling layer.
2. Subsequent layers are convolutional and organised in pairs, owing to the incorporation of residual connections.
3. The number of filters in these layers doubles at each stage, starting from 64. At each stage, since the number of filters doubles, the first convolutional layer in this new stage has the number of filters that is half the number in the previous stage.
4. The network concludes with an average pooling layer, followed by a softmax function.

The design of ResNet-34 effectively addresses the degradation problem commonly encountered in deep neural networks. It is important to note that the ResNet-34 model is pre-trained on the ImageNet dataset, which comprises over 100,000 images spanning 200 distinct classes. This pretraining enables the model to develop a comprehensive understanding of various image categories, thereby enhancing its performance in image classification tasks.

### 3.4.8 Average Pooling Layers

An average pooling layer is a crucial component in neural networks (CNNs), specifically designed for downsampling and reducing the spatial dimensions of feature maps.

In this layer, a sliding window traverses the input feature map, and for each window position, the average value of the elements within the window is calculated. This operation is applied independently to each channel of the input feature map, resulting in a downsampled output feature map. The sliding window's size is determined by the specified height ($P$) and width ($Q$), and a stride ($S$) defines the step size of the window as it moves across the input.

Mathematically, the average pooling operation can be expressed as:

$$O(i,j,c) = \frac{1}{P \times Q} \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} I(i \cdot S + p, j \cdot S + q, c)$$

here, $O(i, j, c)$ represents the value in the output feature map at position $(i, j, c)$, and $I(i \cdot S + p, j \cdot S + q, c)$ denotes the value in the input feature map corresponding to that location.

The usage of average pooling is particularly beneficial in CNNs for several reasons, including its ability to reduce spatial dimensions, control overfitting, and maintain translational invariance, making it a widely adopted technique in modern deep learning architectures.

# 4 Methodology

This chapter provides a comprehensive overview of the methodology employed and the tools we developed to answer the research questions presented in Section 1.3.

Figure 3 illustrates the workflow from initial data acquisition to the development of the predictive machine learning model. This diagram is sourced from a paper written in the context of the WheatVIZ project (Chang-Brahim et al. 2023). In this section, our focus includes the following components: cleaning, preprocessing, and organisation of UAV data (upper section of Figure 3); creation of the dataset (upper right section of Figure 3); and implementation of supervised machine learning techniques (right side of Figure 3).

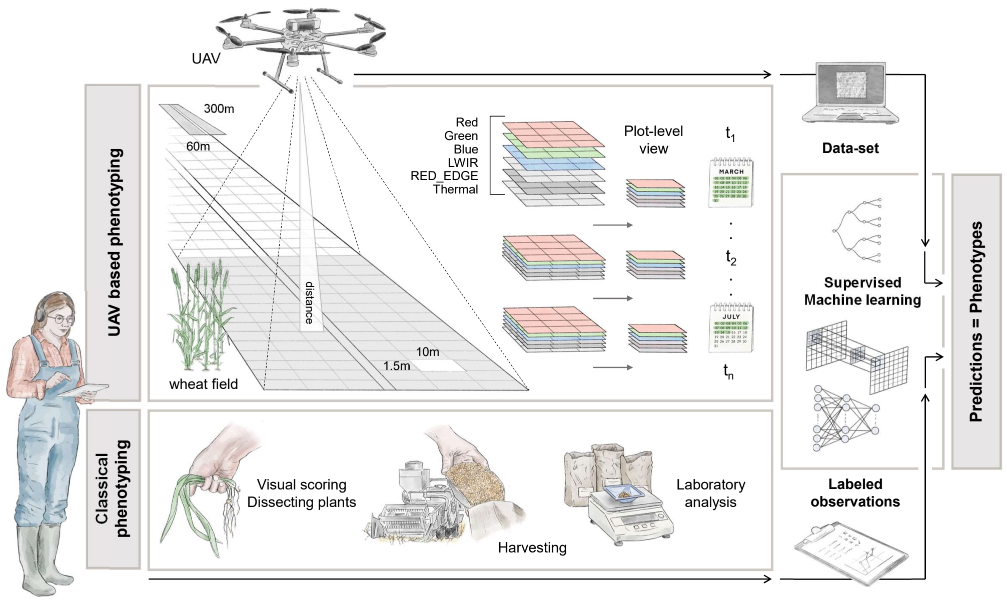


Figure 3: Figure from Chang-Brahim et al. 2023. This figure represents the whole workflow for this research project. It illustrates the passages from the in situ measurements in the field to the deployment of the machine learning models.

## 4.1 UAV Data Collection

In this project, a custom Hexacopter[8], based on a modified Tarot 690 Pro frame using a Cube Orange flight controller running Ardupilot, is deployed. This device can carry heavy loads such as a multispectral camera. The camera used is a 0.5 kg Altum model and acquires RGB, NIR, LWIR and panchromatic images. The UAV maintains a flight

[8]A hexacopter is an unmanned helicopter having six rotors

altitude of 60 meters, resulting in a spatial resolution of 2.5 centimetres. This choice is instrumental in answering the second research subquestion (see 1.3).

The multispectral data were acquired between March 1st and July 11th, 2023. Table 3 reports the measurement dates.

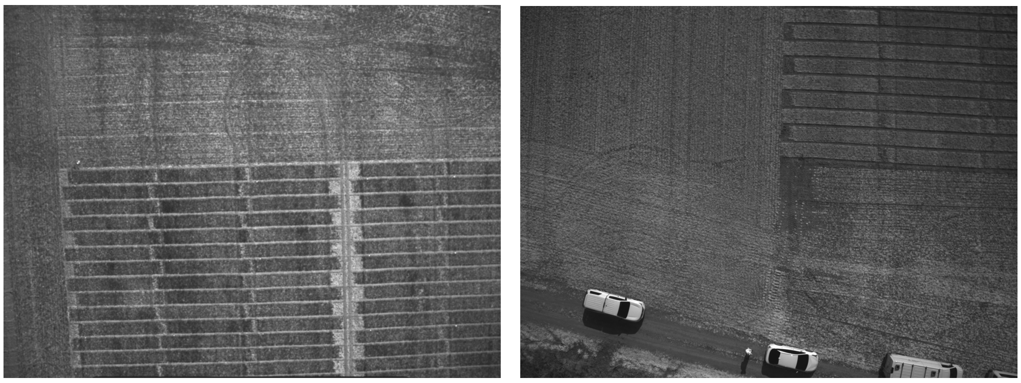
Table 3: Dates of data acquisition.

| Date of the flight |
|---|
| March 22, 2023 |
| April 18, 2023 |
| April 27, 2023 |
| May 15, 2023 |
| May 24, 2023 |
| June 5, 2023 |
| June 14, 2023 |

We initially planned to conduct a flight each week during May 2023, the most important month for the yellow rust emergence for wheat plants (Chen et al. 2014). Unfortunately, due to heavy rainfall in May, acquiring multispectral data weekly, as planned, became unfeasible. Drone flights were hindered by the weather, a common limitation of UAV-based technologies. Additionally, the drone sustained damage after the June 14th flight, requiring a three-week repair period, resulting in a missed opportunity for subsequent measurements.

AIT's UAV systematically flies over the agricultural field. Equipped with a camera capturing RGB, thermal, long-wave infrared (LWIR), panchromatic, and red-edge measures (see Table 1) during each flight.

Two images acquired by the drone are presented below in Figure 4 as a reference.



(a) Survey of plot lines. (b) Image acquired during lift-off.

Figure 4: Panchromatic drone acquisition images captured on April 18th.

## 4.2 Feature Engineering: from Reflectance Maps to Plot-level Data

Our research involves partitioning the experimental field into georeferenced plots, with domain experts collecting phenotypic data. This data serves as the ground truth for all the machine learning approaches in this thesis. Having in situ phenotypic data represents an uncommon practice due to its cost but holds great value for the machine learning pipeline (Nguyen et al. 2023, Tang et al. 2023). The grid-like structure aligns with established practices in various research studies, ensuring consistency with conventions in controlled crop cultivation conditions (Haghighattalab et al. 2016; Bai et al. 2016; Volpato et al. 2021). Figure 3 visually outlines the grid structure and individual plots.

Using Pix4D photogrammetry software, we processed the collected data to produce a reflectance map[9] (for details see Section 4.2.1). The quality of the map primarily relies on the extent of pixel overlap in the images. Typically, maintaining 5 or more images per pixel serves as a reliable threshold for quality (Pix4D Support 2024). In Figure 5 it is clear that the field area is covered by 5 or more images per pixel.

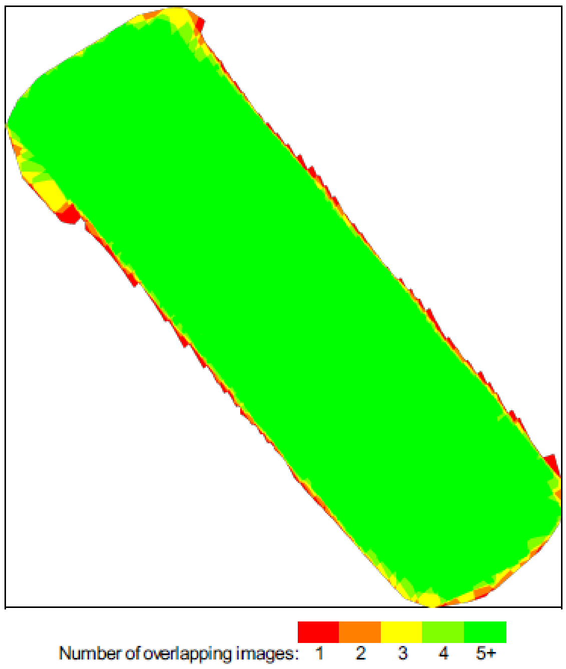


Figure 5: Orthomosaic pixel overlap: Red/yellow areas indicate low overlap, decreasing the quality of the resulting pixels. Green areas have over 5 images per pixel, ensuring good quality with sufficient key point matches (see Stitching in Section 4.2.1).

[9]A reflectance map illustrates the varying reflectance properties across the field

Before we describe the creation of the dataset that is employed in the machine learning part, two passages are relevant: stitching the images together (Section 4.2.1) and extracting the plot-level information (Section 4.2.2).

### 4.2.1 Orthomosaic Creation

Pix4D mapper has emerged as one of the most prominent tools for creating orthomosaics (see Section 2.4). In this thesis, we employ the Pix4D mapper (version 4.8). It features advanced photogrammetry algorithms that ensure accurate image stitching, resulting in precise orthomosaics and 3D models. Furthermore, Pix4D mapper offers robust georeferencing capabilities, aligning the outputs with the desired real-world coordinates for future integration into a Geographic Information Systems (GIS). In particular, it can generate accurate reflectance maps and enable practitioners to manipulate them as raster images (Rasmussen et al. 2016).

Each set of multispectral images is processed by Pix4D following the following processing steps (Beltrame et al. 2024, Pix4D Manual 2024):

- Image matches: align spatial features in adjacent images in the surveyed area. This process identifies key points, i.e. common points between two images that match. The images are not merged yet, since they need to be orthorectified and radiometrically and geographically corrected.
- Geolocation: assign geographic coordinates to each pixel in the image for precise spatial referencing. This involves determining the camera's position and orientation during capture relative to Ground Control Points (GCPs). Through triangulation using GCPs' known ground positions and their corresponding positions in the images, Pix4D estimates the location of every pixel, ensuring accurate alignment with real-world coordinates.
- Orthorectification: correct for terrain-induced distortions due to different terrain heights. This step ensures an accurate representation of features on the earth's surface. Project the image onto a planar surface using a Digital Elevation Model (DEM) to account for terrain relief effects.
- Mosaicing/stitching: combine multiple orthorectified images to create a cohesive orthomosaic of the study area. This process blends pixel values from overlapping images for a consistent representation of the landscape.
- Calibration: calibrate the camera to make the sensor function as accurately as possible, ensuring uniformity across all images acquired in the same flight. Radiometric calibration, a critical component, ensures consistency in pixel values by correcting for sensor response, atmospheric conditions, and illumination differences. It also utilises reference targets of known reflectance, such as

metal plaques, to establish a correlation between sensor output and surface properties, which is then applied to all pixels in the image.

Among these steps, the most innovative for airborne-acquired agriculture data is the incorporation of Ground Control Points (GCPs). GCPs are chessboard-like plaques positioned on the ground and precisely geolocated using a GNSS. An example of a GCP is reported in Figure 6.

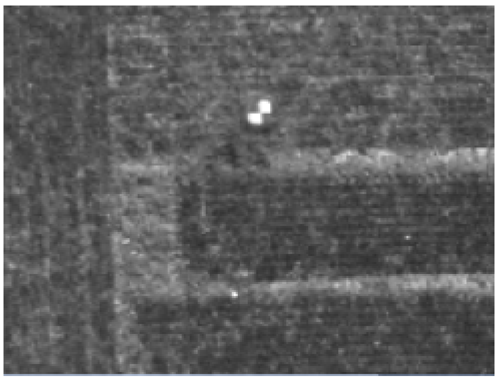


Figure 6: A close-up view of GCP in the field, guiding the way to accurate georeferencing. The GCP is the chequerboard-like plate.

This thesis underscores their significance in enhancing pixel geolocation in the results section 5.1. We outline the disparity between the original georeferenced images and those corrected post-GCP implementation.

In Appendix E, we also present the punctual procedure to create the radiometrically calibrated orthomosaic using Pix4D. This procedure is applied to all 7 flights we acquired in the measurement campaign.

In Figure 7 a reconstructed orthomosaic, displaying the NDVI is presented as a reference.

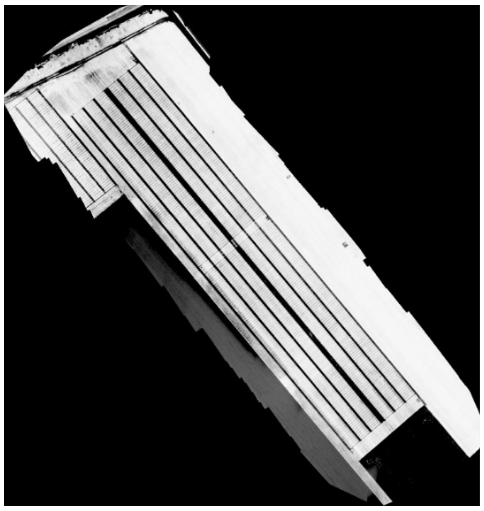


Figure 7: This high-resolution image depicts the NDVI over the experimental field. The acquisition height is 60 m while each plot is 1.5 m x 10 m. The brighter areas represent vegetation, while the darker ones represent soils or the road.

### 4.2.2 Plot-level Information Extraction

Subsequently, Edelhof's team utilises high-precision GNSS geolocalised reference points, which correspond to the extremal points of each field plot, to generate binary image overlays that extract the pixels belonging to each plot. This step proves to be necessary because we need to identify univocally the contour of each plot.

As a reference, a depiction of the overlays in Obersiebenbrunn is presented in Figure 8.

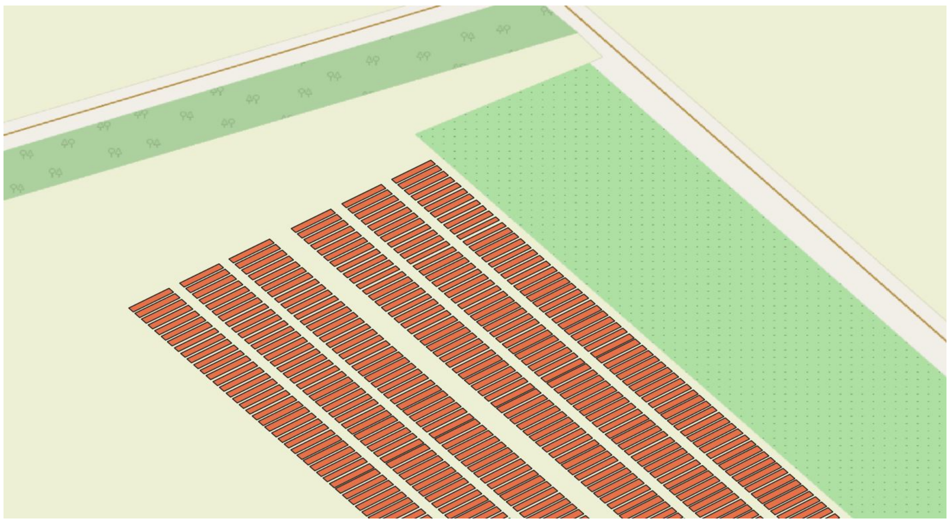


Figure 8: Overlays created using the data provided by the Edelhof team. The underlying map is provided by Open Street Map and represents the facility in Obersiebenbrunn. Each orange quadrilater is a single plot-level overlay.

The agriculture field on which the overlays are applied is presented in Figure 9:

The reflectance maps are then used to retrieve the plot-level data for each plot. We retrieve reflectance maps' pixels beneath the overlays, obtaining a small portion of non-zero pixels (corresponding to the plots) fully embedded in zeroes, since the overlays introduce them. Those images are then trimmed to reduce their size, ensuring that the outermost non-zero values touch one of the borders of the images. As the plot orientations are not aligned, adjustments are necessary, since the overlays introduce zeros, potentially causing artifacts or distortions in the image data (Hashemi 2019). These artefacts could hinder machine learning algorithms by adding unnecessary information or altering spatial relationships within the image. Ensuring images are rectangular can prevent these issues, improving machine learning model accuracy (Belcher et al. 2023). Hence, we reoriented the images and conducted resampling to ensure precise image reconstruction, as detailed in Section 3.2.

An example of a resampled image for the RGB band is Figure 10.

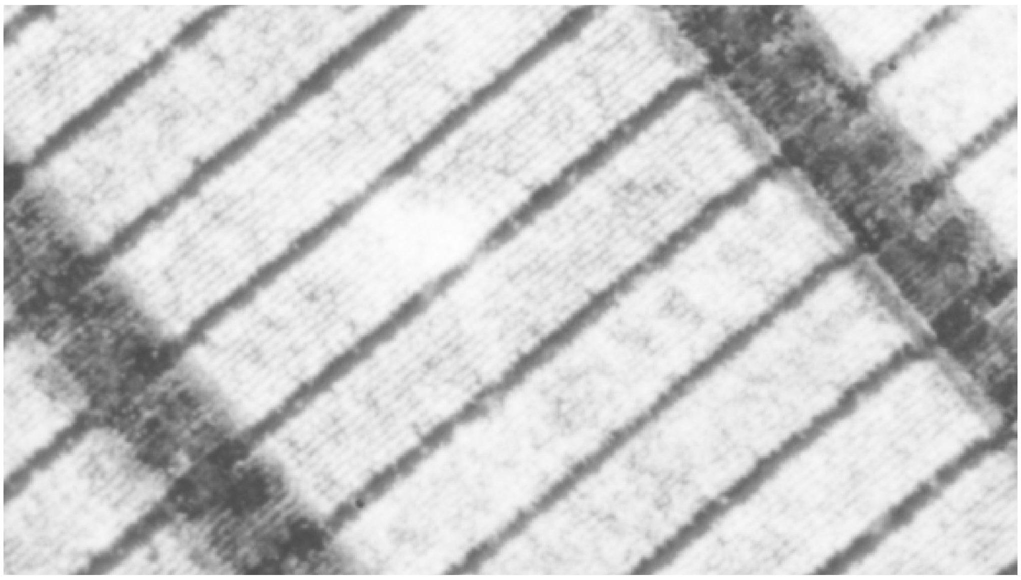


Figure 9: Detail of the field acquired at the operational height of 60 m in the panchromatic band. The image had already been orthonormalised, calibrated and readjusted. Each plot is 1.5 m x 10 m.

Various resampling filters, such as interpolation algorithms and nearest neighbour approaches, were considered to resample the images when applying the projective transforms to reorient the images (see Section 3.2.1). In particular, we selected four resampling procedures: the nearest neighbours (NN), bilinear, bicubic and Lanczocs (for additional information see Section 3.2.2).

In the process of selecting the most appropriate interpolation method for image resampling, a thorough analysis of our available options was conducted in Section 5.2. The primary objective was to strike a harmonious balance between the preservation of details of the image, the minimising of potential artefacts and ensuring seamless transitions between pixels. Bicubic interpolation consistently emerged as a robust performer, effectively retaining fine image details.

While Lanczos interpolation yielded results similar to bicubic, it was observed to have a slow processing speed, as evidenced in Figure 21. Conversely, the nearest neighbour and bilinear methods, at times, introduced undesirable irregularitied, as visible in Figure 19.

It is worth emphasising that our decision-making process extended beyond quantitative assessments presented in Section 5.2. Qualitative visual inspections by a photogrammetrist played a significant role in our evaluation, allowing for the assessment of overall image quality and clarity. Moreover, an in-depth analysis of pixel-level details was undertaken to ensure the chosen interpolation method aligns closely with

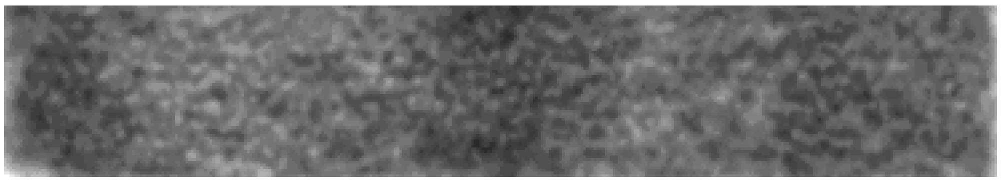


Figure 10: A panchromatic figure showcasing a resampled plot captured on April 27, 2023, from an altitude of 60 meters. The image, reconstructed using a bicubic kernel, measures 372 pixels in width and 64 pixels in height, representing real-world dimensions of 10 meters by 1.5 meters.

our objective of preserving image fidelity. This comprehensive approach reinforces the rationale behind our selection of bicubic interpolation as the preferred method, given its ability to effectively balance image quality and computational efficiency.

Once plot images have been extracted and resampled in each spectral band for every conducted flight (refer to Table 3), we restructure this data to yield 7448 tensors, one for each plot and flight. In subsequent sections, these will be referred to as "multispectral cubes". This data is organised in a time-series manner, with multispectral cubes associated with the same plot merged and ordered by acquisition date (see Table 3). The result is a dataset comprising 1064 elements, each an entry containing a 4-dimensional tensor consisting of acquisition date (referred to as time step from now on), spectral channel, width, and length. The time series is composed by 7 timesteps. The channels have a dimensionality of 7 (panchromatic has been dropped since it is useful only to calibrate the RGB images and in further analysis would be co-linear with RGB information since it is their composition). Width has a dimensionality of 64 and length has a dimensionality of 372.

### 4.2.3 Two-dimensional Tabular Data Preparation

Two-dimensional tabular data is essential for training and utilising various basic machine learning models, including linear models, support vector machines (SVM), and random forests. Before describing the models employed, we survey how we prepared the data for the basic machine learning approaches.

To obtain a tabular dataset from the multispectral cubes we gather data associated with each plot and calculate pixel distribution statistics for each channel, which includes mean, standard deviation, kurtosis and the 25th, 50th, and 75th percentiles. This results in a vector of six statistics for each spectral channel, totalling a dimensionality of 42. This resulting vector is associated with the disease score related to its plot (see Section 1.2 and Appendix A). The choice to prioritise easily calculable global descriptors over the usual local descriptors in image analysis stems from the decision to create simple baseline models using basic machine learning (refer to Section

4.4) and to use them to evaluate the performance of machine learning approaches (refer to Section 4.5). This tabular data representation is a recurring theme in our basic ML approaches, aimed at simplifying and streamlining spectral data analysis. This feature design choice proved insufficient to predict yellow rust abundance using the basic models. A more refined version of this, which includes time information, constitutes an interesting direction for further research.

However, it is worth noting that in the more advanced deep learning section, we implicitly incorporate both local and global descriptors (as a reference see Figure 13). We achieve this by employing convolutional layers and pre-trained models, which excel at capturing fine-grained details within images, alongside pooling layers for data aggregation. This combined approach allows for a balanced extraction of both local and global information, providing a comprehensive representation of the data.

To summarise, we use tabular data for the basic models. The choice of not including time information determines problems for the basic models, as explained in Section 6.1. Nevertheless, in more complex deep learning methods, intricate local descriptors are employed, but the chosen network design remains centred on aggregating pixel-level information (see Section 4.5).

## 4.3 Experimental Set Up

In this section, we explain how we record the machine learning results of the experiments conducted, the managing of the models' hyperparameters and how we split the dataset.

### 4.3.1 Experiment Tracker and Configuration System

In this thesis, experiments are meticulously parameterised through exhaustive configurations, represented by YAML files. Each experiment possesses its unique "experiment config," detailing how to override the overarching "default config" for reproducibility. The "default config" is used to provide a comprehensive list of all the possible models' parameters and hyperparameters. We create the default configuration with the idea of managing both training experiments for basic ML models and also for deep learning models. We also include the experiment configuration to perform evaluation experiments and data preprocessing. This way, we keep track of all the operations we performed and ensure the complete reproducibility of the experimental results. This hierarchical parameter overwriting is a commonplace practice across various config management libraries[10]. Specifically, we adopt the YAECS library (Golubev 2024). Using open-source config systems such as this makes the scientific process:

[10]Examples of configuration system managers include YACS, OmegaConf and HYDRA, widely used for automating and managing configuration and deployment in IT environments.

- more robust by implementing safeguards against typos and errors.
- more efficient by making it easy to start new experiments and keep all previous results accessible.
- more ethical by helping make each experiment distinct, meaningful, and trivial to reproduce.

In this thesis, we also leverage ClearML (ClearML 2016), a commercial tracker renowned for its intuitive interface for manipulating and storing experiment results. ClearML systematically organises and documents research progress, simplifying the monitoring of changes, tracking of experiments, and management of experiment results. When combined with YAECS, it also ensures reproducibility and validation by maintaining comprehensive records of parameter and hyperparameters values and enabling the efficient fetching of experiment results. Additionally, it fosters collaboration by providing a transparent platform for sharing and discussing work.

### 4.3.2 Dataset Split: Balancing Training, Validation, and Test Sets

In this section, we introduce a method for splitting the datasets defined in Sections 4.2.1 and 4.2.3. As each dataset instance corresponds uniquely to an experimental plot, we propose a split of the plot IDs, which consequently divides the dataset instances. For brevity and clarity, we term this process 'Dataset Splitting'. We decide not to perform split in the time configuration because we want to use the full information carried by the time series, since each time step refers to a different stage of development of the plant.

To evaluate the performance of the trained models, we use both one validation and three test sets. Therefore we organise the plots into five distinct groups (one training set, one validation set and three test sets), based on Edelhof's experimental design in Obersiebenbrunn. The rationale behind the decision to employ 3 test sets is to explore whether the model can generalise well also on plots containing plants with different phenological properties, such as a different irrigation status or a different breed (or genotypes).

The field experiments performed by Edelhof categorise plots into various groups. Each plot is characterised by three codes:

- **Replication:**
  - "Replication 1" signifies a normal experiment plot.
  - "Replication 2" designates the control group plot.
- **Water State:**
  - "W" denotes the absence of artificial irrigation.

  - "W0" indicates the presence of artificial irrigation.

- **Wheat Breed/Genotype:** The wheat variety used in a particular plot is denoted by the letter "B"[11] followed by a number representing the year in which the genetic material of the seed was acquired:
  - "B1": Refers to a wheat breed selected for the year 2021.
  - "B2": Refers to a wheat breed selected for the year 2022.
  - "B3": Refers to a wheat breed selected for the year 2023, the most recent one.

  Incorporating wheat varieties from previous generations used in 2022 and 2021 enables breeders to assess the yield and characteristics of the current generation under different weather conditions as those experienced in prior years. Additionally, when using the same variety, breeders can evaluate how a refined selection of seeds (the breed selected in 2023) from the same lineage (the breed selected in 2022 or 2021) would perform in the same weather conditions, allowing for a comparative analysis of field results from previous years in standardised conditions.

It is important to notice that each breed of wheat has 4 plots, two per irrigation state and 2 per replication, and that our data contains seeds selected across several past years.

For clarity, a plot obtained from seeds selected in the 2022 generation, with artificial irrigation and being used as a replication will be identified with the following: [Replication 2, W0, B2] or, in short, 2-W0B2. Similarly, the replication 1 plot is identified by 1-W0B2.

Figure 11 is a schematic representation of plot data in the field experiments in Obersiebenbrunn. It contextualises the previously mentioned characteristics.

[11]B is a shorthand to replace the scientific name of the breed of wheat used. For example, B can be substituted with "SU Habanero" or "SE 432-22", two different breeds of winter wheat.

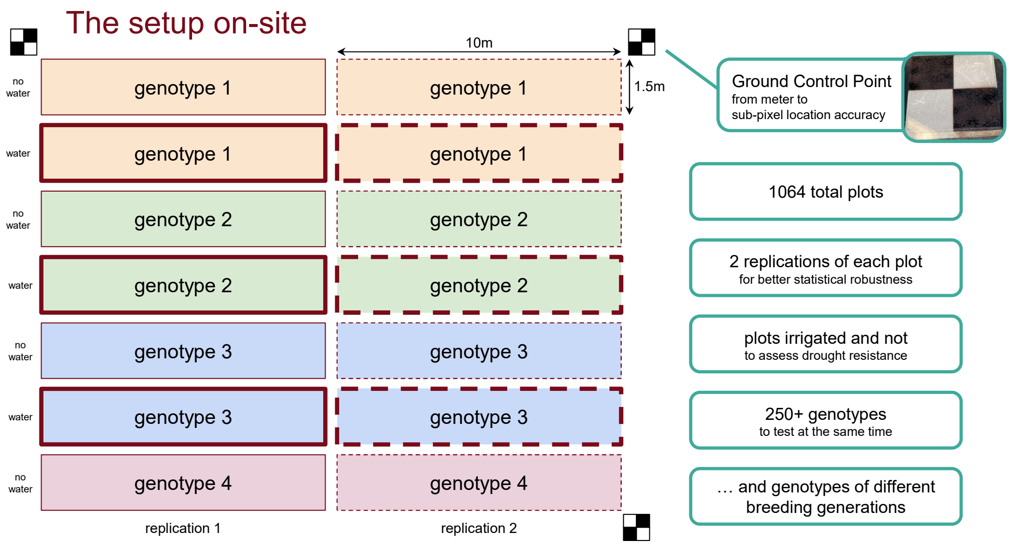


Figure 11: Organisation of experimental trials conducted in Obersiebenbrunn. Each breed of wheat has 4 plots, two per irrigation state and 2 per replication. This image is sourced from Beltrame et al. 2024.

We establish three distinct test sets, each associated with different plant traits. We constitute test sets 2 and 3 from our plots associated with breeds selected in 2021 (and only those), resulting in a total of 40 plots. These are then divided into two sets based on water status, yielding two sets, each comprising 20 plots. The identifiers for these sets follow the format: 1-W0B1 and 2-W0B1 for test set 3, 1-WB1 and 2-WB1 for test set 2, where 'B' represents ten possible breeds from the generation selected in 2021. Both sets include replication 1 and replication 2 plots, serving to evaluate the model's performance in predicting phenotypes for plants originating from seeds selected in previous years under different water regimes.

Subsequently, we randomly sample 10 genotypes from the remaining plots, encompassing seeds selected for the years 2023 and 2022, as well as replication 1 and 2, and both irrigated and non-irrigated cases, to evaluate the models' performance on unseen breeds (or genotypes). The plots in this set are identified by: 1-W0B3, 1-W0B2, 1-WB3, 1-WB2, 2-W0B3, 2-W0B2, 2-WB3.

To establish training and validation sets, we leverage the inherent structure of the remaining data, particularly its replication plots. With a limited dataset comprising only 984 instances, we aim to maximize training data while ensuring robust model performance. Recognizing the scarcity of data, we allocate 50% of replication 2 samples to the training set alongside the replication 1 plots, comprising plots identified by: 1-W0B3, 1-W0B2, 1-WB3, 1-WB2, 2-W0B3, 2-W0B2, 2-WB3, and 2-WB2. By incorporating replication 2 data in our training set, we seek to enhance the model's robustness and generalization capabilities despite the small dataset size. The remaining data constitute the validation set, comprising plots identified by 2-W0B3, 2-W0B2, 2-WB3, and 2-WB2. In this approach the validation set is intended as a set to confirm that the model is able to predict data similar to the training set and to perform hyperparameter tuning independently from the training set.

Table 4: Distribution of data split for machine learning. In test set 1 $\hat{B}$ stands for breeds that were not previously selected. There is no overlap between breeds/genotypes represented by $\hat{B}$ and B.

| **Split** | **Cardinality** | **Description** | **Identifiers** |
|---|---|---|---|
| Train | 738 | Training set comprising plots identified by replication 1 and 2, both irrigation states, and various wheat breed samples | 1-W0B3, 1-W0B2, 1-WB3, 1-WB2, 2-W0B3, 2-W0B2, 2-WB3, 2-WB2 |
| Validation | 246 | Validation set consisting of replication 2 samples, both irrigation states, and different wheat breed samples | 2-W0B3, 2-W0B2, 2-WB3, 2-WB2 |
| Test 1 | 40 | Test set with plots associated with 10 opted-out breeds. Plots are replication 1 and 2, subject to both irrigation states, and different wheat breed samples | 1-W0$\hat{B}$3, 1-W0$\hat{B}$2, 1-W$\hat{B}$3, 1-W$\hat{B}$2, 2-W0$\hat{B}$3, 2-W0$\hat{B}$2, 2-W$\hat{B}$3, 2-W$\hat{B}$2 |
| Test 2 | 20 | Test set for evaluating model performance on seeds selected in 2021 in a non-irrigated state | 1-WB1, 2-WB1 |
| Test 3 | 20 | Test set for evaluating model performance on seeds selected in 2021 in an irrigated state | 1-W0B1, 2-W0B1 |

This rigorous train test split is instrumental in answering the research question "How can remote sensing image data be transformed and aggregated to extract characteristics for predicting disease scores?" since we propose a way to aggregate the remote sensing data (defined in Sections 4.2.1 and 4.2.3) based on the Edelhof field design to create a framework to conduct rigorous machine learning experiments. Moreover, it also contributes to the question "In the context of evaluating the viability of aerial machine learning-based phenotyping as an alternative to traditional in situ phenotyping, how can a machine learning model be designed to validate its performance, leveraging domain-specific agricultural knowledge?" since having a train-test-validation split is one of the best practices in Deep learning. Moreover we made design choices, in particular the inclusion of replication 2 instances in the training set, to mitigate the problem of data scarcity. In Figure 49 we present the disease scores distribution for the training and the validation sets. In Figure 50 we present the disease scores distribution for the three test sets.

### 4.3.3 Loss Function and Evaluation Metrics

During training, we employ MSE (see Section 3.3) as the loss function. In the regularised version of the basic algorithms, as detailed in Sections 3.4.1 and 3.2.2, and in the deep learning models, as elaborated in Sections 4.5.5 and 4.5.6, we also incorporate the regularisation terms.

As the evaluation metrics, we use the Mean Absolute Deviation (MAD, see Section 3.3.2), and the $r^2$ score (see Section 3.3.3). Additionally, we introduce the adjusted $r^2$ score, specifically selected for deep learning models due to its capability to address concerns related to model complexity and the number of predictors.

## 4.4 Basic Machine Learning Models

Basic machine learning models are not computationally intensive when compared to most deep learning approaches on small datasets, as in our case. As explained in Section 3.4, Grid Search Cross-validation (CV) is a rigorous technique for hyperparameter selection for basic machine learning models. Even if there are more sophisticated approaches to hyperparameter selection, such as Bayesian search, we decided to seek simplicity and efficiency and optimising further the hyperparameter selection for baseline models would be outside the scope of this thesis.

In addition to this, for the basic machine learning model, we chose to use the procedure detailed in Section 4.2.3 to convert the time series of multispectral cubes (see Section 4.2.1) into a tabular format. To do so, we select only the components of the time series associated with the day 24th of May, the day of disease score acquisition. The rationale behind this choice is that, even though deep learning approaches use sequential data, the basic ML models constitute the first simple approach and therefore we decide to constraint the dataset. At the beginning of the experiments, we thought that the acquisition date was the most important day for our regression problem. Nevertheless, this assumption will be revealed to be incorrect, since, as detailed in Section 5.5.4, the deep learning procedure selects the dates after the 24th of May as the ones that contribute the most to the model performances. Nevertheless, this choice serves its role in kickstarting our analysis.

We start our exploration of machine learning models with linear regression. Given its simplicity and minimal hyperparameters, it serves as an initial point of reference. Following this, we delve into regularised versions of the linear model and introduce SVMs. We conclude with the random forest model.

### 4.4.1 Linear Models

We explore three model variations: the standard linear model and two regularised counterparts. The Ridge regressor (see 3.4.2) is employed to alleviate multicollinearity[12] by shrinking all coefficients towards zero (Kennedy 2003). Additionally, the lasso regressor (see 3.4.1) is considered, promoting sparsity by encouraging specific feature coefficients to be precisely zero. The only hyperparameter we fine-tuned for the regularised models is the strength of the regularisation ($\alpha$). For both the lasso and the ridge the constant is selected between the values reported in Table 5.

Table 5: Values of the $\alpha$ hyperparameter for lasso and ridge regressors.

| $\alpha$ | **Values** |
|---|---|
| Lasso | 0.001, 0.01, 0.1, 1, 10 |
| Ridge | 0.001, 0.01, 0.1, 1, 10 |

### 4.4.2 Support Vector Machine

Support Vector Machines (SVMs), especially their kernelised variations, present versatile and powerful modelling tools for creating simple yet effective models. To address the non-linearity inherent in the aggregated image data (refer to Section 4.2.3), we initially employed the classical SVM, as detailed by the optimisation problem in Equation 2. Subsequently, we explored kernelised versions, specifically utilising the Radial Basis Function (RBF) and sigmoid kernels. While we also experimented with polynomial kernels of second and third orders, their run times quickly became impractical, prompting a focus on the more efficient RBF and Sigmoid kernels. This methodology aims to leverage the flexibility of SVMs, especially in handling non-linear relationships in the high dimensional space given by the statistics of the images.

In practice, we employed the scikit-learn[13] implementation of this algorithm (Pedregosa et al. 2011). For the bandwith of the kernel we use the default setting of scikit-learn. During the tuning of the hyperparameters, we focused on the following configuration:

- **Kernel:** The type of kernel function used, in particular, linear, polynomial, Gaussian and sigmoidal.
- **C:** The regularisation hyperparameter. A smaller C encourages a smoother decision boundary, while a larger C allows for a more complex decision boundary

[12]Multicollinearity occurs when independent variables in a regression model are highly correlated, making it challenging to discern their individual effects on the dependent variable

[13]We use scikit-learn version 1.4.0

that closely fits the training data. The values selected for this approach are: [0.001, 0.01, 0.1, 1, 10].

- **Degree:** For polynomial kernels, this hyperparameter represents the degree of the polynomial. Only the polynomials of second and third orders are considered for CV.

### 4.4.3 Random Forest Regressor

Random forest is a well-established model. Its ensemble nature creates a robust model to get better predictions than previous linear models. In this thesis, we employed the scikit-learn implementation and fine-tuned a range of hyperparameters to optimise model performance. These hyperparameters include:

- the number of decision trees in the forest (N Estimators).
- the maximum depth of individual trees (Max Depth).
- the minimum number of samples required for a split (Min Samples Split).
- the minimum number of samples in a leaf node (Min Samples Leaf).
- the maximum number of features considered per split (Max Features).
- the maximum number of leaf nodes in each tree (Max Leaf Nodes).
- the maximum number of samples used in each tree (Max Samples).
- the decision whether a node splits based on the impurity decrease (Min Impurity Decrease).
- the decision whether the bootstrapped sampling techniques are used (Bootstrap ).

The hyperparameters presented in Table 6 provides ample flexibility for tailoring the Random Forest model making it an adaptable and powerful tool for addressing complex regression challenges.

Table 6: Hyperparameters and values for the RF CV.

| **Hyperparameter** | **Values** |
|---|---|
| N Estimators | 10, 20, 50, 100, 200, 500 |
| Max Depth | 3, 6, 10, 20, 30, 50, 100, None |
| Min Samples Split | 2, 5, 10, 20 |
| Min Samples Leaf | 1, 2, 4, 10 |
| Max Features | 1, auto, square root of samples, log2 of samples |
| Max Leaf Nodes | 2, 5, 10, 20, 50, 100, None |
| Max Samples | 0.5, 0.7, 0.9, 1.0 (i.e.; no bootstrap sampling) |
| Min Impurity Decrease | 0, 0.01, 0.05, 0.1, 0.5 |
| Bootstrap | True, False |

## 4.5 Deep Learning Approaches

In this section, we provide an overview of deep learning architectures and their associated methodologies, with a specific emphasis on elucidating the underlying reasons behind the design decisions.

### 4.5.1 Data Augmentation

In Section 4.2.1, we discussed the preprocessing steps involved in transitioning from raw data to well-structured multispectral images. In Figure 12 it is possible to see a visual representation of an instance of the dataset.

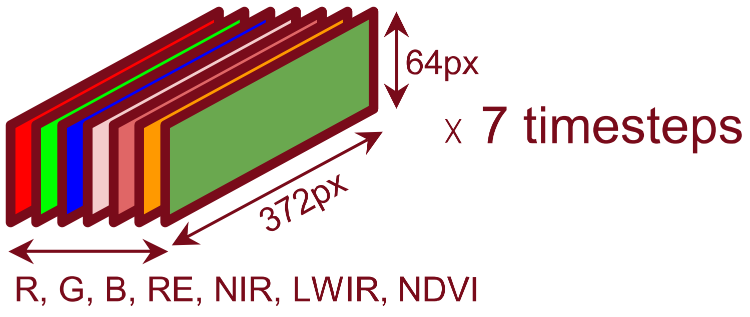


Figure 12: Schematic representation of the deep learning inputs. R stands for Red, G for Green, B for Blue, RE for Red Edge, NIR for Near Infrared, LWIR for Long Wave Infrared and NDVI is the Normalised Difference Vegetation Index. The seven timesteps refer to the 7 flights, as illustrated in Table 3.

The training images undergo a min-max normalisation, re-scaling them to the interval [0, 1] (see as detailed in Section 3.2.3). The min-max normalisation is applied independently to each time step and each channel. For each time step and each channel, we compute the maximum and the minimum value of the pixel over the width and length dimension and we use them to normalise the image. As a result, the algorithms' gradient descent can more effectively navigate the loss landscape, facilitating the convergence of the model during training (Hestness et al. 2017).

During each epoch, the multispectral cubes are subject to data augmentation. Data augmentation strengthens the model's robustness and helps to learn invariants to generalise better on unseen data, by employing diverse transformations, such as image flips. Thereby, it helps mitigate the pervasive risk of overfitting and improving adaptability to real-world scenarios (Rebuffi et al. 2021). Additionally, data augmentation plays a role in balancing class distributions and optimising the usage of limited data (Ottoni et al. 2023). Augmentation in a standard procedure in ML, hence we perform it.

In the data augmentation process, we have incorporated a set of transformations to enhance the richness of our dataset. These transformations include horizontal flipping, vertical flipping, the introduction of random noise, and the application of salt and pepper noise. The latter is a technique where random pixels are either completely black (pepper) or completely white (salt), simulating noise and further diversifying the dataset. We also acknowledge the existence of numerous other augmentation techniques within the temporal domain (Wen et al. 2020), and we remain open to exploring these possibilities in future iterations of our work.

For the following formulas, we adopt the convention that $x$ and $y$ represent the coordinates of a pixel in the original reference system, while $x'$ and $y'$ denote the transformed pixel. $W$ and $H$ stand for the width and height of the image.

- **Horizontal Flipping:** This transformation involves reflecting an image horizontally. Mathematically, it is described as:

$$x' = W - x \quad \text{and} \quad y' = y$$

- **Vertical Flipping:** This operation mirrors the image along the vertical axis and can be expressed as:

$$x' = x \quad \text{and} \quad y' = H - y$$

- **Random Noise:** The introduction of random noise simulates small data variations. Moreover, it also combats overfitting in cases where the Gaussian noise has no real-world interpretation and might stabilise training by preventing the model from becoming overconfident which would cause it to fall into local minima (Shorten and Khoshgoftaar 2019). This technique involves adding random values to the pixel intensities of the image. Mathematically, it can be expressed as:

$$I'(x,y) = I(x,y) + N_{\sigma,0}$$

  where $N_{\sigma,\mu}$ is Gaussian noise with standard deviation $\sigma$ and mean $\mu = 0$, and $I(x,y)$ are the pixel values in the original image and $I'(x,y)$ in the modified one.

- **Salt and Pepper Noise:** This technique introduces random pixels with either complete black (pepper) or complete white (salt) intensities. Mathematically, it can be expressed as:

$$I'(x,y) = \begin{cases} 0 & \text{with probability } p_{\text{p}} \cdot p_{\text{s \& p}} \\ 1 & \text{with probability } p_{\text{s}} \cdot p_{\text{s \& p}} \\ I(x,y) & \text{else} \end{cases}$$

here, $p_\mathrm{p}$ and $p_\mathrm{s}$ represent the probabilities of introducing pepper and salt pixels, respectively. $p_{\mathrm{s\,\&\,p}}$ is the probability of performing the salt and pepper augmentation. $I'(x, y)$ is the transformed value of the pixel and $I(x, y)$ is the original value. This additional transformation contributes to the robustness of the model by introducing different types of noise and variations into the dataset.

These augmentations are sequentially applied, each with an independent probability. Consequently, their effects can accumulate. The rationale behind stacking transformations lies in the idea of exposing the model to a more comprehensive set of variations within the data. For instance, horizontal and vertical flipping can simulate changes in orientation or perspective, while the introduction of random noise replicates the inherent uncertainties or variations present in real-world data. By combining these transformations, we aim to capture a broader spectrum of potential variations, enabling the model to generalise better and perform well on unseen data by learning to recognise different perspectives and variations, ultimately improving its ability to handle different scenarios during inference. Table 7 presents an overview of the probabilities relative to each augmentation.

Table 7: Table representing the probabilities that a data instance is augmented with the respective transformations.

| Augmentation | Probability |
|---|---|
| V-flip | 0.5 |
| H-flip | 0.5 |
| Salt and Pepper ($p_{\mathrm{s\,\&\,p}}$) | 0.3 |
| Gaussian | 0.3 |

We tested different augmentation probabilities, but we settled on this configuration for efficiency's sake. We assigned a 0.5 probability to the flip transformation as it doesn't alter the image. Additionally, we refrained from using higher values for salt and pepper augmentation to avoid excessively prolonging the convergence epoch. A more structured approach to this problem might be investigated in the follow-up research of this work.

### 4.5.2 Convolutional Neural Network

In this thesis, we use two convolutional neural networks as the feature extractors of the model (see Figures 13 and 14): the first is a simple stack of convolutional layers (see Section 3.4.5), while the second is ResNet34 (see Section 3.4.7). In this final approach, this pre-trained CNN model leverages multispectral transfer learning to alleviate data scarcity (He et al. 2015). The main challenge that arises, in this case, is how to convert 2-dimensional CNNs to offer a practical solution for effectively fusing information from different spectral bands to reveal valuable insights within 8-channel multispectral images across a range of applications. An in-depth discussion of this approach can be found in Section 4.5.6.

### 4.5.3 Average Pooling Layer

Average pooling layers (see Section 3.4.8) are employed to reduce the size of feature maps in the deep learning pipelines (see Figures 13 and 14). In the case of our investigation into yellow rust abundance, the disease score depends on the collective contribution of plants within an experimental plot, where the count of diseased plants influences the agronomist's assessment. To mirror the real-world disease score acquisition as closely as possible, we opt to integrate an average of the feature map instead of a max pooling layer.

This decision stems from a concern that a max pooling layer might yield disproportionately negative scores in cases where a small batch of diseased plants coexists with an otherwise healthy population, thus neglecting the contribution of the latter. We believe that employing an average would offer a fairer representation of the overall health of the plot.

Due to time constraints, we are unable to explore the potential impact of including max pooling layers in our model within the scope of this thesis.

### 4.5.4 Long Short-Term Memory

As we discussed in Section 3.4.6, LSTMs excel at capturing time dependencies within the data. We Since the data employed in this study are image data, the features to be fed to the LSTM cells need to be extracted from the images themselves. In this case, the feature maps extracted by the CNN, after an averaging pooling layer (see Section 3.4.8), might constitute a prime input. Regarding the output of the model, an LSTM network outputs a sequence of hidden states, each representing the network's internal memory and encoding relevant information about the input sequence in each time step.

On this premise, we can transition to our first deep learning setup.

### 4.5.5 CNN-LSTM Network

This is the first model among the ones proposed that considers explicitly both the spatial and time dependency inherent in our data. In the context of this thesis, this is a deep learning baseline that would serve as a benchmark for the more refined approach proposed in Section 4.5.6. Nevertheless, this architecture can predict phenotypes, even though it does not generalise well, as explained in Section 5.4. Figure 13 presents the schematic representation of the model. The hyperparameters of this architecture can be found in Appendix B.1, in Table 36.

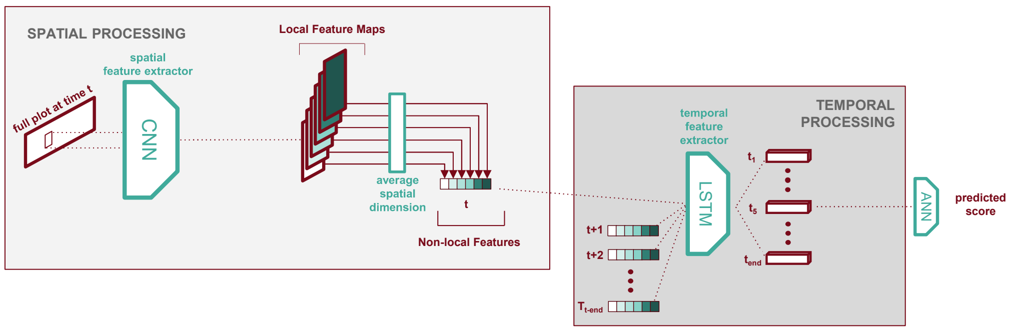


Figure 13: Illustration depicting our custom CNN-LSTM model architecture. In the initial phase, a feature extractor (stacked convolutiona layers in this case) is employed to capture spatial information from each plot at every time step. Subsequently, the rigth hand side leverages the abundant time information available to enhance the model's temporal understanding.

The model can be divided into two parts: CNN extracting spatial features and an LSTM combining those features over different timesteps. The feature extractor is based on an adjustable number of convolutional layers (the CNN in Figure 13). The spatial feature extractor takes as input all bands for a plot at a single time step, which is a tensor image with shape [C, W, H], where C is the number of channels (7) and W and H are respectively the width and length. The adaptable number of channels was implemented to make it possible to have a scalable architecture to make it possible to add new spectral bands and indices in future iterations of this work.

When the input tensor passes through M convolutional layers, the shape of the output feature map (filters in Figure 13) is calculated using the procedure presented in Section 3.4.5. This procedure takes into account the local features and patterns of the multispectral images. To get a more compact representation of the spatial information, an average pooling layer (see the description is 3.4.8) is added after the convolutional layers to downsample the feature map thus reducing its dimensionality while retaining essential information, therefore obtaining non-local features. The pooling layer (average spatial information in Figure 13) operates with fixed-size rectangular windows that compute the mean value within the window independently for each channel in the feature map. The result is a feature map reduced to a single element with unchanged channel depth (raw extracted features in Figure 13). This downsampling process reduces the risk of overfitting, leading to a feature map with shape [C, 1, 1], where C is the number of channels. To implement it practically we used the standard PyTorch implementation, using the torch.nn.AvgPool2d module. This module automatically computes the features to downsample to the desired width and height of the resulting feature map.

After this procedure is applied to all the multispectral cubes, the features have shape [L, C, 1, 1], where L is the number of time steps in the time series.

The feature maps are then used as the input of an LSTM cell (LSTM in Figure 13). The assumption and experimental ansatz here is that the network until this point can learn the spatial and spectral features well enough to be able to generalise. From this point onwards in the network, we will be using the LSTM to model the temporal dependencies. As described in subsection 3.4.6, the output of the LSTM is a tensor with shape [L, C]. In this LSTM, we only consider a single timestep, therefore the final output of this model would be a tensor of shape [C]. This is a fixed-size representation of the time dependencies between the extracted spatial features of all preceding timesteps.

The number of LSTM modules is a tunable hyperparameter. Moreover, the activation of each LSTM, excluding the last one is subject to a ReLu layer and a drop-out layer before the next LSTM module. This approach helps mitigate overfitting.

Finally, an Artificial Neural Network (ANN in Figure 13) is added to control the model's output more efficiently and to apply dimensionality reduction to get a single scalar as the output. The number of hidden layers and the number of neurons are tunable parameters. Similarly to the LSTM case, ReLu and dropout layers are inserted between the hidden layers to mitigate overfitting.

### 4.5.6 ResNet-LSTM Network

This represents our most comprehensive model for addressing the prediction of the yellow rust phenotype. Figure 14 illustrates the model architecture, replacing the CNN layers in the feature extractor with a pre-trained ResNet network (He et al. 2015). We chose ResNet for its simplicity compared to more modern networks (e.g., Tang et al. 2023) and its ability to capture spatial features crucial for our application. The approach prioritizes efficiency and generalisation over higher-resolution methods. ResNet's effectiveness in crop disease prediction has been demonstrated in previous studies (Zhang et al. 2019).

In practice, we utilised the feature extractor module of ResNet34, optimising the depth of the feature map by analising the "index_out" hyperparameter (see Section 3.4.7 and the timm library timm 2024).

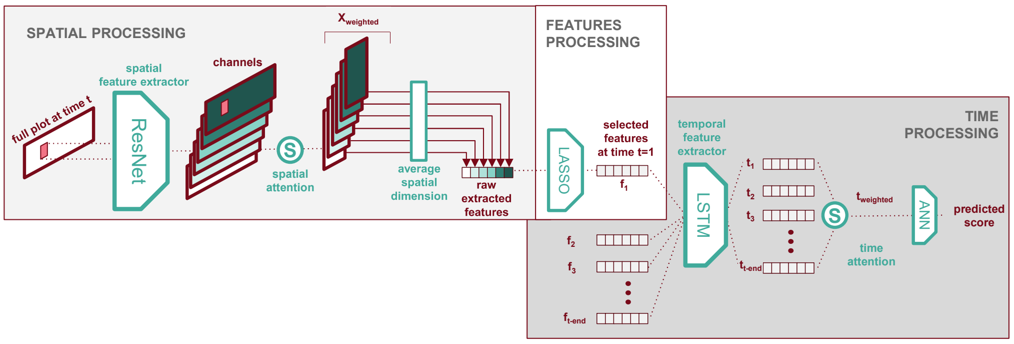


Figure 14: Illustration depicting the custom ResNet-LSTM model architecture. On the left-hand side, a feature extractor (ResNet) captures spatial information from each plot at every time step. Subsequently, on the right-hand side, it leverages the time information contained in the time series. The way spatial attention works is presented in Figure 16

.

Each data instance is subject to a preprocessing step that, for each timestep, divides the initial 7-channel multispectral cube into three 3-channel cubes, as described in Figure 15. The flow of the network then involves partitioning each multispectral cube (described in Figure 15) into chunks along the width dimension, resulting in $C_{Chunks}$ chunks of 3 channels. The rationale behind this choice is that partitioning the plot areas into sub-plots would take into account different areas with different distances from the edges of the plot. This is intended to account for microclimate variations of the plant biome, between the plants close to the external environment and plants deep into the plot. Each component is subsequently independently processed by ResNet, and the subsequent feature maps undergo batch normalisation and ReLU before being merged into a single feature representation using the spatial attention mechanism detailed in Figure 16.

After extracting spatial features, we perform an average over each resulting channel of the feature map on the width and height dimensions (see "average spatial dimension" in Figure 14). Subsequently, a L1 regularised linear layer selects important features, creating a representation $f_i$ for each time step. This process is repeated for each time step, and the resulting tensors are fed into an LSTM, providing a new feature representation considering the intrinsic time dependencies in the data. This representation undergoes batch normalisation, ReLU and then a weighted average attention mechanism, merging all time information into a single representation $t_{weighted}$ (see Figure 17 for details), which is then fed into an ANN to obtain disease scores.

**Data preparation**

Before delving into the specifics, we discuss a further process of data preparation. Pre-trained models are typically trained on RGB images, and this presents a challenge when working with multispectral data. The fundamental issue lies in the inability to directly deploy these models for processing multispectral data due to their unique characteristics. To overcome this challenge, we have developed another data preprocessing approach. Each multispectral cube is equipped with 7 distinct channels. We grouped our input channels 3 by 3 to create tensors compatible with our pre-trained model's input shape. Figure 15 provides a schematic representation of the process.

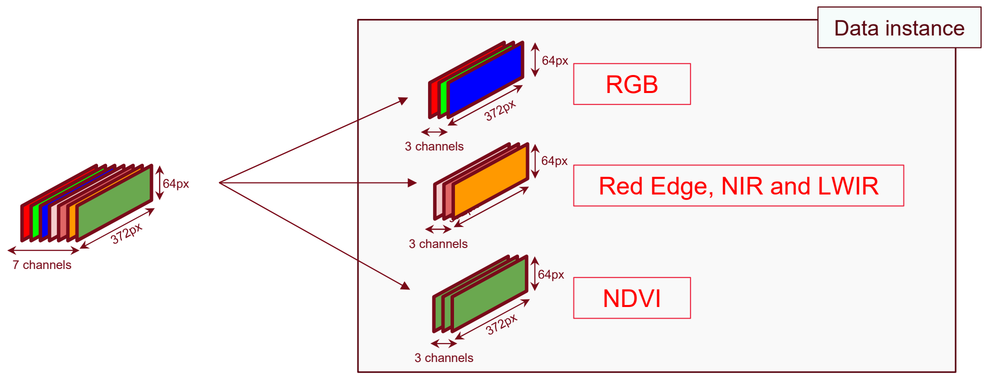


Figure 15: Preprocessing step to get 3 channel multispectral cube for the ResNet-LSTM network.

The three resulting tensors are:

- **Standard RGB Cube**: this tensor retains the original RGB cube (see upper multispectral cube in Figure 15). The model can effectively generalise patterns and discern underlying insights from this standard cube.

- **Aggregated Long Wave Multispectral Cube**: in this tensor, we amalgamated Near-Infrared (NIR), Long-Wave Infrared (LWIR), and red-edge measures into a single multispectral cube (see middle multispectral cube in Figure 15). This aggregation was motivated by the longer wavelengths of visible light in these measures, which share similar physical properties. Incorporating this cube enhances the ability of the model to capture thermal properties and related patterns.

- **NDVI-Enhanced Multispectral Cube**: this tensor is composed of three replicates of a Normalised Difference Vegetation Index (NDVI) raster image (see bottom multispectral cube in Figure 15). By tripling the NDVI representation, we over-represent this index due to its significance in agriculture.

Essentially, our stratification process restructures multispectral data into three tensors, each finely tuned to convey distinct physical information to the feature extractor. As far as the author knows, this is a novel adaptation and it enables a model pre-trained on RGB data to employ multispectral data by creating pseudo-RGB cubes. To comprehensively study this phenomenon an ablation study is necessary, but this goes beyond the scope of this thesis.

Each multispectral cube of the triplets presented in Figure 15 undergoes an additional normalisation as per ResNet's specifications. Specifically, the three channels of each cube are re-normalised using the values reported in Table 8.

Table 8: ResNet-specified mean and standard deviation values for each channel, originally corresponding to RGB bands but in our application representing multispectral cubes (see 4.5.6).

| | Channel 1 (R) | Channel 2 (G) | Channel 3 (B) |
|---|---|---|---|
| Mean | 0.485 | 0.456 | 0.406 |
| Standard deviation | 0.229 | 0.224 | 0.225 |

**Regularisation and attention mechanisms**

To improve the model's performance and to make the model determine which spatial and temporal components are the most important, we incorporate two regularisation strategies. Those utilise weighted average attention mechanisms. The first one is a spatial attention mechanism, while the second one employs a time attention mechanism. Figure 16 provides a schematic representation of the spatial attention procedure while Figure 17 is a representation of the time attention.

For each time step, instead of directly feeding each multispectral cube into the feature extractor, we divide the input into a predefined number of chunks $C_{Chunks}$.

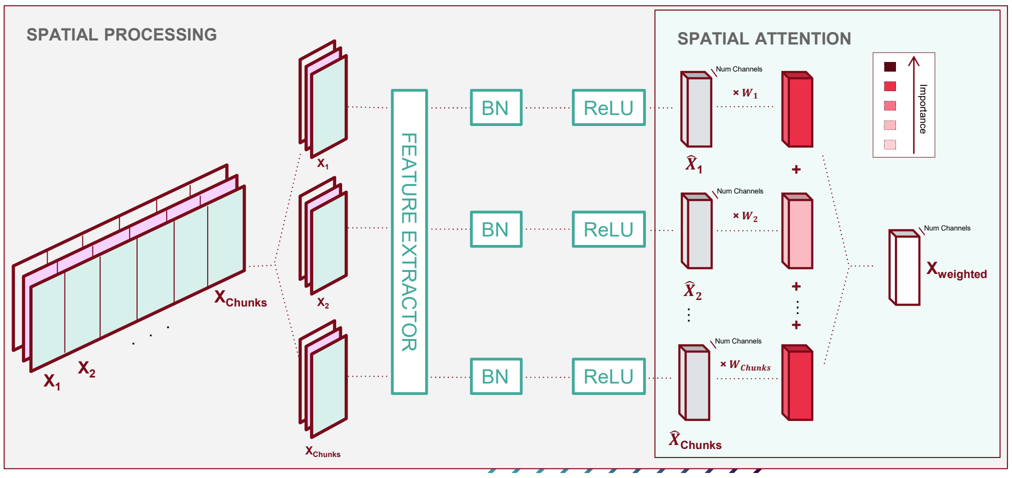


Figure 16: Spatial attention module in the spatial processing pipeline. This procedure is applied to each channel and for each spectral band, spatially selecting the most relevant spatial information. BN refers to the Batch Normalisation module and ReLU is the Rectified Linear Unit. The tensor is represented on the left side of the figure in any of the three presented in Figure 15. All three multispectral cubes are processed independently from the feature extractor.

Then, these fragments are passed through the ResNet feature extractor ($ResNet_{FE}$) to produce feature maps. Each feature map undergoes weight averaging to create a consolidated spatial representation to assign weights based on their significance. The weights used in this operation were then subjected to L1 regularisation and integrated into the loss function to encourage sparsity. This process can be mathematically represented as follows:

$$x_{weighted} = \sum_{i=1}^{C_{Chunks}} w_i \cdot \text{ReLU}(\text{BN}(\text{ResNet}_{FE}(x_i))) = \sum_{i=1}^{C_{Chunks}} w_i \cdot \hat{x}_i \tag{3}$$

where $x$ is the input data passed through the feature extractor (ResNet$_{FE}$), followed by Batch Normalisation (BN) and Rectified Linear Unit (ReLU) activation. The resulting $C_{Chunks}$ feature maps are then combined using weights ($w_i$) derived from the attention mechanism. These weights are applied to the feature maps and the final weighted sum ($x_{\text{weighted}}$) is obtained.

The normalised weights $w$, also known as importance scores, are normalised using a softmax function, hence, for each component:

$$w_i = \frac{e^{\hat{w}_i}}{\sum_{j=1}^{n} e^{\hat{w}_j}} \tag{4}$$

where $w_i$ is the output probability for the $i$-th element, $\hat{w}_i$ is the $i$-th element of the attention weights vector, and $n$ is the number of elements in the input vector. In the spatial attention n is the number of chunks ($C_{Chunks}$) or the number of time steps (T) in the time attention

The same procedure detailed in Equation 3 is applied to the feature representation ($t_{weighted}$) that aggregates all the time information. Figure 17 provides a schematic representation of the time attention procedure.

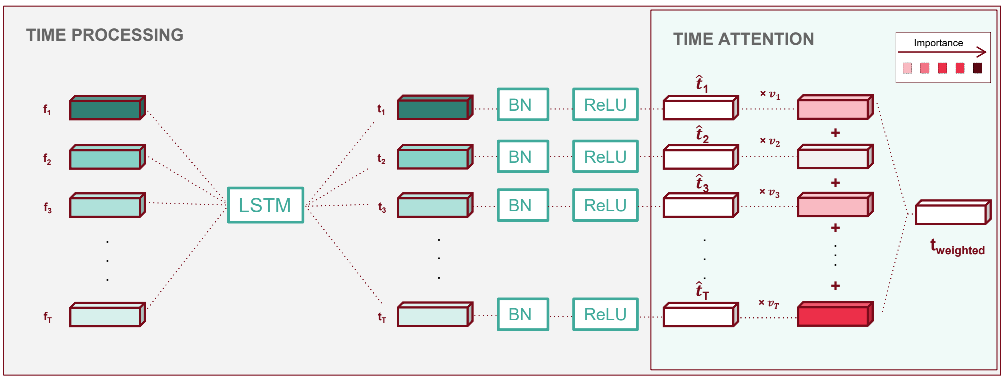


Figure 17: Time attention module in the time processing pipeline. This module is applied to each spatial feature representation output by the spatial processing pipelines. The attention mechanism selects which timesteps of the time series contribute the most to the model predictions. BN refers to the Batch Normalisation module and ReLU is the Rectified Linear Unit.

Equation 3 can be rewritten as follows:

$$t_{\text{weighted}} = \sum_{i=1}^{T} v_i \cdot \text{ReLU}\left(\text{BN}\left(t_i\right)\right) = \sum_{i=1}^{T} v_i \cdot \hat{t}_i$$

where $t_i$ represents the temporal features processed through Batch Normalisation (BN) and Rectified Linear Unit (ReLU) activation. The resulting feature vectors, denoted as $\hat{t}_i$, are then combined using weights ($\mathbf{v}_i$) obtained from a temporal attention mechanism. These weights are applied to the feature tensors and the final weighted sum ($t_{weighted}$) is calculated by summing over all $i$ from 1 to $T$.

This procedure lets the model itself determine which element of the time series contributes the most to the prediction problem after its spectral and spatial information are aggregated by the spatial and feature processing modules (see Figure 14). This procedure improves its ability to recognise relevant patterns and reduces the impact of redundant information.

### Cost Function

As for the cost function, sparsity is introduced by adding the spatial regularisation term to the original mean Squared Error (MSE) cost function ($J_0$). This modified cost function ($J'$) is given by:

$$J' = J_0 + \lambda \sum_{i=1}^{C_{Chunks}} |w_i| \tag{5}$$

where $\lambda$ controls the strength of L1 spatial regularisation, $w_i$ is the i-th element of the spatial attention weights $\mathbf{w}$, and $J'$ is the new cost function.

Similarly, building on Equation 5, the cost function is expressed as:

$$J_{Attention} = J' + \gamma \sum_{i=1}^{T} |v_i| = J_0 + \lambda \sum_{i=1}^{C_{Chunks}} |w_i| + \gamma \sum_{i=1}^{T} |v_i| \tag{6}$$

here, $\gamma$ governs the strength of L1 time regularisation, $v_i$ is the i-th element of the time attention weights $\mathbf{v}$, and $J_{Attention}$ is the cost function incorporating spatial and temporal attention.

The final cost function incorporates regularisation by the feature selector over weights $\phi_i$ (lasso in Figure 14):

$$J = J_{Attention} + \Omega \sum_{i=1}^{T} |\phi_i| = J_0 + \lambda \sum_{i=1}^{C_{Chunks}} |w_i| + \gamma \sum_{i=1}^{T} |v_i| + \Omega \sum_{i=1}^{T} |\phi_i| \tag{7}$$

here, $\Omega$ controls the strength of L1 feature regularisation, $\phi$ represents time attention weights, and $J$ is the final cost function combining three regularisation techniques.

Extending this to include L2 regularisation introduced with weight decay, the cost function becomes:

$$J_{Final} = J + \beta \sum_{j=1}^{N_{Params}} (\Lambda_j)^2 \tag{8}$$

here, $J_{Final}$ incorporates attention, L1 feature regularisation, and weight decay. The hyperparameter $\beta$ controls the strength of weight decay, $\Lambda_j$ is the j-th trainable weight parameters of the DL model, and $N_{Params}$ is their the total number.

# 5 Results

In this section, we examine the outcomes of our research, evaluating the performance of the various models utilised, including SVMs, linear models, and RFs, with a specific emphasis on deep learning models. Additionally, we will visualize the learning curves for our ML models. It is essential to acknowledge that our research encountered limitations stemming from diverse data-related issues, and we will discuss these limitations alongside our findings.

## 5.1 Impact of Ground Control Points

Ground Control Points (GCPs) significantly enhance the precision of geolocation. Figure 18 illustrates the difference between pixel coordinates derived solely from GNSS data, i.e.: the image that would result by skipping the GCP part in the procedure described in Section 4.2.1, and those refined through triangulation with GCPs. Each column in Figure 18 denotes the average difference between the geographical coordinates in each of the seven flights performed.

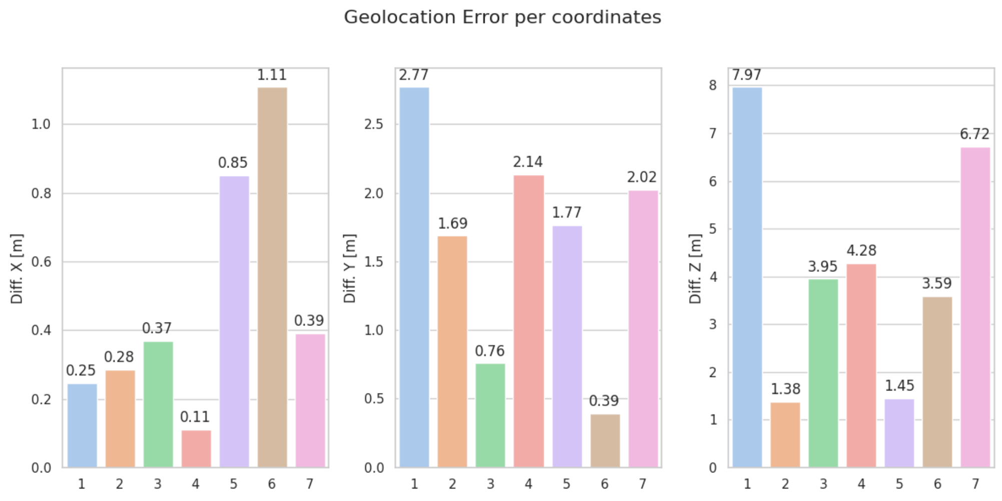
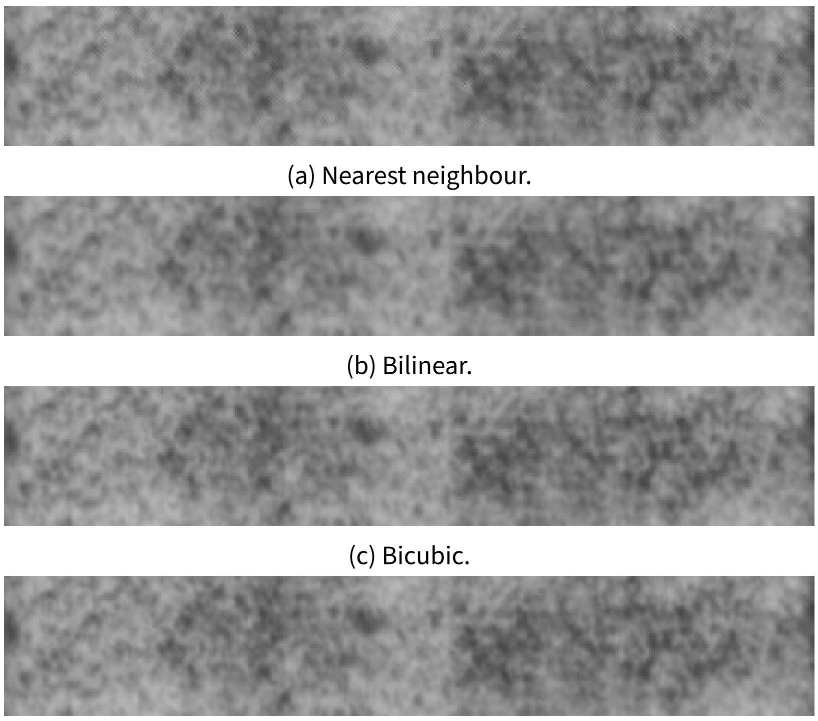


Figure 18: Barplot depicting the average error between all the coordinates that are not corrected with GCPs and the corrected ones for each orthomosaic obtained. The numbers on the x-axis correspond to the flights in Table 3. Notice the scale of the y-axis changes in the three different plots.

The contribution of GCPs plays an important role in data quality assessment. In fact, given that the plots are 1.5 meters wide, an error of 1 meter could misidentify one plot from another, potentially compromising data reliability by incorrectly geolocalising one pixel, with all its associated information to the wrong plot. Moreover, the correction on the Z axis is useful to correct the obtained terrain model, in the "Mosaicing" passage of the procedure in Section 4.2.1. The usage of GCPs significantly mitigates this effect by obtaining sub-pixel geolocation errors in the 3 directions.

## 5.2 Orthomosaic Reconstruction Error

In Figure 19, we present an example of reconstructed images using various resampling algorithms. The quality of the reconstruction is easily appreciable for both the bicubic and the Lanczocs kernels[14], while nearest neighbour and bilinear kernels do not reconstruct well the underlying image details.


(a) Nearest neighbour.

(b) Bilinear.

(c) Bicubic.

(d) Lanczos.

Figure 19: Plot images reconstructed with different resampling techniques for the same NDVI acquisition.

Figure 20 illustrates a benchmark analysis comparing resampling algorithms. Mean Squared Errors (MSEs) were calculated for each kernel (nearest neighbours, bilinear, and bicubic) considering an image reconstructed with them and one reconstructed

[14]The difference in the reconstructed images between these two kernels has been validated qualitatively by Phillipp Fanta-Jende, a photogrammetrist, even though it is present only in very small details, such as in the right-hand side of the plot.

with the Lanczocs resampling. MSEs and runtimes are averaged across the reconstruction of 1064 images for each of the 7 channels considered. The normalisation of MSEs allows for easier comparison, particularly considering the difference between the wavelength of the RGB band (400 - 495 nm) and the LWIR band (8000 - 14000 nm). This determines different orders of magnitude in the errors between pixel values and, therefore we can not compare these two bands. To solve this problem, all MSEs are divided by the variance of the pixels in the original plot image (i.e.; before the projective transform described in Section 4.2.2) in the chosen channel. This procedure helps to present the data more effectively and makes the comparison possible.

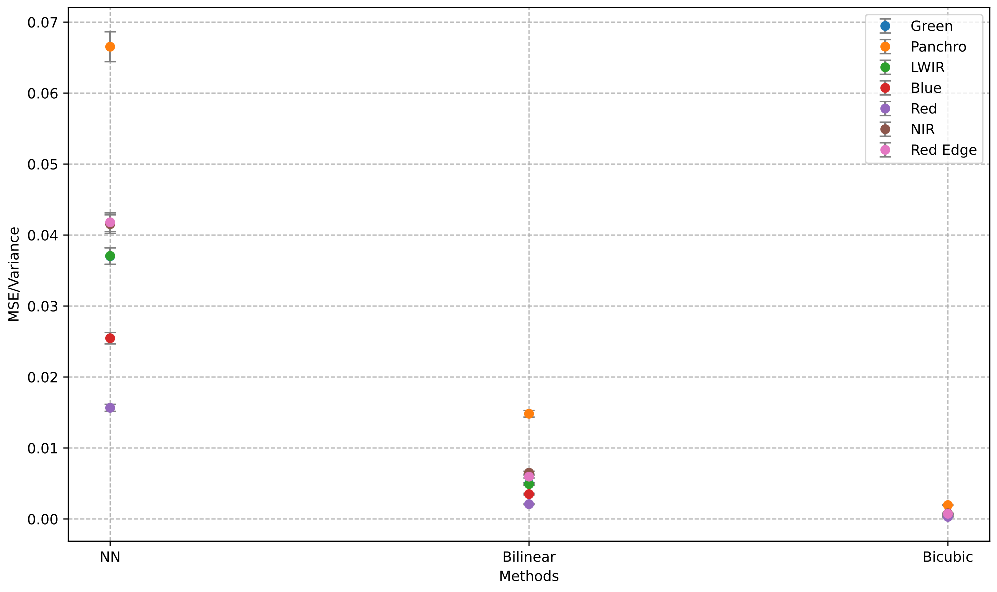


Figure 20: Comparison of images resampled with different methods with the Lanczos resampled image. Mean Squared Errors are normalised to account for variance, acknowledging the wavelength difference in orders of magnitude between RGB and the LWIR band. The error bars are computed using the standard error of the mean.

Resampling processes exhibit varying runtimes. In our setup, average runtimes for reconstructing 1064 multispectral images are detailed in Figure 21.

The observed outcomes align with expectations. While Lanczos resampling displayed superior accuracy, the incremental benefit over Bicubic did not justify the doubling of runtime. This suggests that considering the scalability of the model, we achieved a favourable trade-off. Conversely, the accuracy of NN and bilinear resampling was not as high, as evident from the reconstructed images and Figure 20.

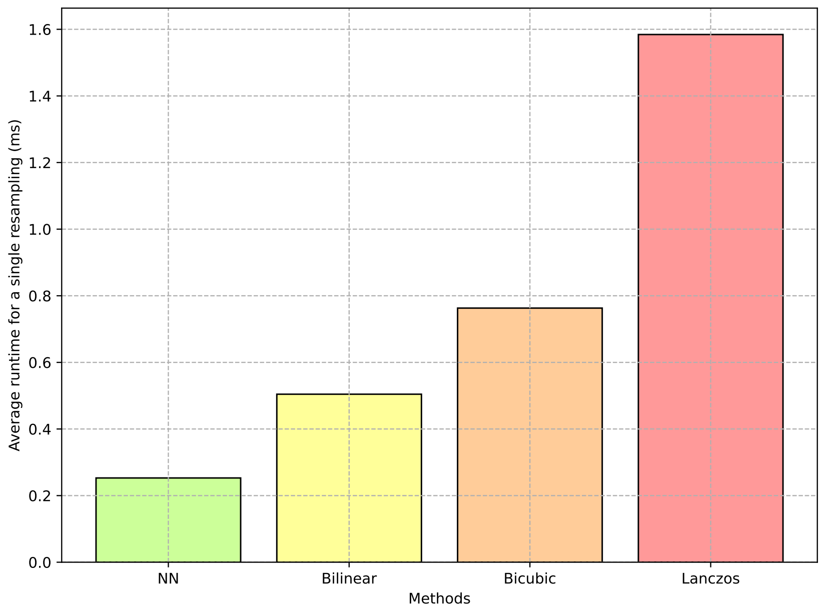


Figure 21: Average runtimes for different resampling algorithms. Results represent an average of 1064 runs, with each run reconstructing a single image.

## 5.3 Basic Machine Learning Models

In this section, we will present the results of the machine learning models we utilise. These models have limitations as, due to the data preprocessing procedure to generate the tabular dataset (see Table 4.2.3), we choose not to exploit the potential of a time series analysis. For all the basic models the hyperparameters are selected using Cross-Validation (CV) using the training set. As previously discussed in Section 4.4, these models serve as baseline models and early trials before the introduction of more complex models. The rationale behind this decision is that we want the basic models to use the same number of training instances as the deep learning approaches (see Section 4.5). This choice contributes to keeping the experimental approach standardised and answering the research question "How can we effectively predict phenotypic traits in crops, particularly focusing on disease scoring, considering the trade-off between model simplicity and complexity?".

Since none of the approaches yielded satisfactory results on the validation set, for the sake of brevity and to avoid redundancy, we did not report the values of the metrics of the three test sets (see Section 4.3.2).

Among the basic models, only the random forest produced satisfactory results on the training set; however, its performance on the validation set is poor (see Figure 26).

### 5.3.1 Linear Model

Linear models, as expected, fell short of generating meaningful predictions due to the inherent challenge of representing multispectral cubes solely through statistical features, as evident in Table 9. In Figure 22 it can also be appreciated how the model substantially predicts the average value of the target for all the instances. Hence the model is not able to predict the higher and lower disease scores.

In reality, the tabular data employed did not reveal any inherent data patterns, resulting in a loss of valuable information. Nevertheless, it holds significance to have a reference point for comparison, even if the linear models did not yield substantial results, as this allows us to contrast our more complex models' performance against the baseline and not against a random model.

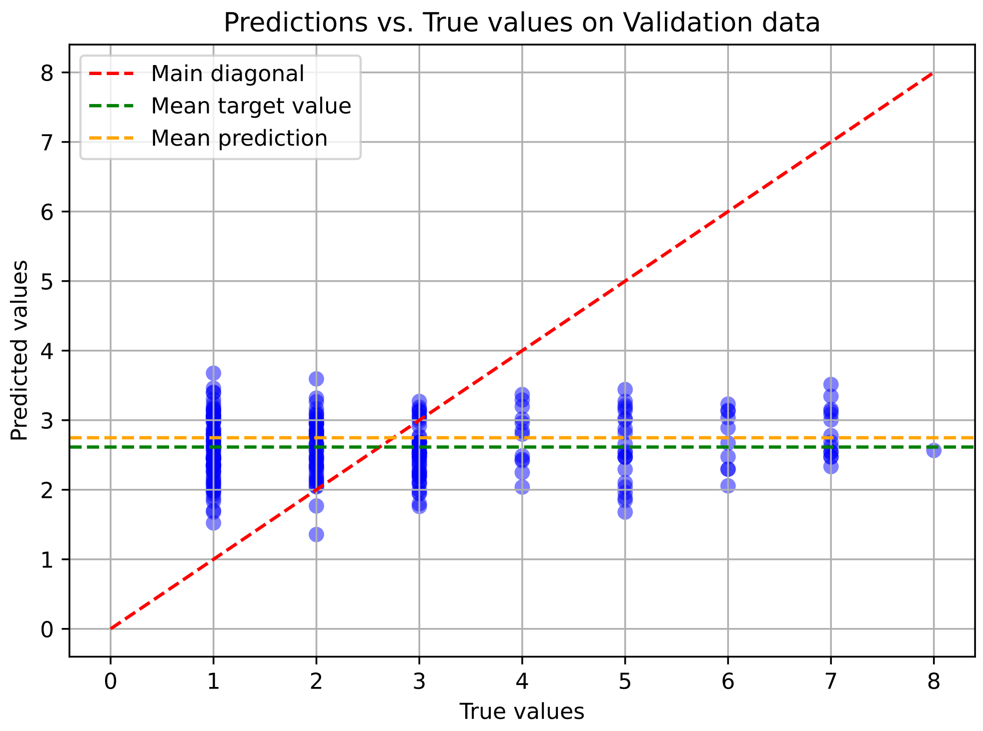


Figure 22: This figure presents the predictions given by the linear model on the validation set. The green dotted line is the average of the disease scores of the validation set, while the orange line is the mean of the predictions for the validation set.

The $r^2$ score suggests that on average the linear model is predicting the average of the data distribution; therefore it is not learning even on training data. We can better appreciate this effect looking at the horiziontal lines in Figure 22.

Table 9: Metrics and CV results for the linear regression model. The hyperparameters of the best model are selected via CV and the metrics on training and validation are computed using them. The CV is performed using 5 folds.

| CV results k=5 | |
|---|---|
| Average MSE | 3.91337 |
| Average $r^2$ | -0.02202 |
| Average MAD | 1.60620 |

| Training set | |
|---|---|
| MSE | 3.71598 |
| $r^2$ | 0.04015 |
| MAD | 1.57500 |

| Validation set | |
|---|---|
| MSE | 3.41317 |
| $r^2$ | -0.01598 |
| MAD | 1.48025 |

The linear model exhibits limitations in predicting disease scores, resulting in bad overall performance. Consequently, metrics were not computed for the test sets.

### 5.3.2 Lasso Regressor

The lasso regressor yield similar results to the previous approach, showcasing that regularisation is only marginally more effective in solving the regression problem. The CV selected $\alpha = 0.01$ as the strength of regularisation. In Table 10 we can appreciate that the cross-validated results are poor, having a model that essentially predicts, on average, the average of the target data. Figure 23 shows that the lasso regression reduced even further the variance of the prediction on the validation set. This brought a marginal increase of the $r^2$ score because the classes with target 1 and 2 are the most populated.

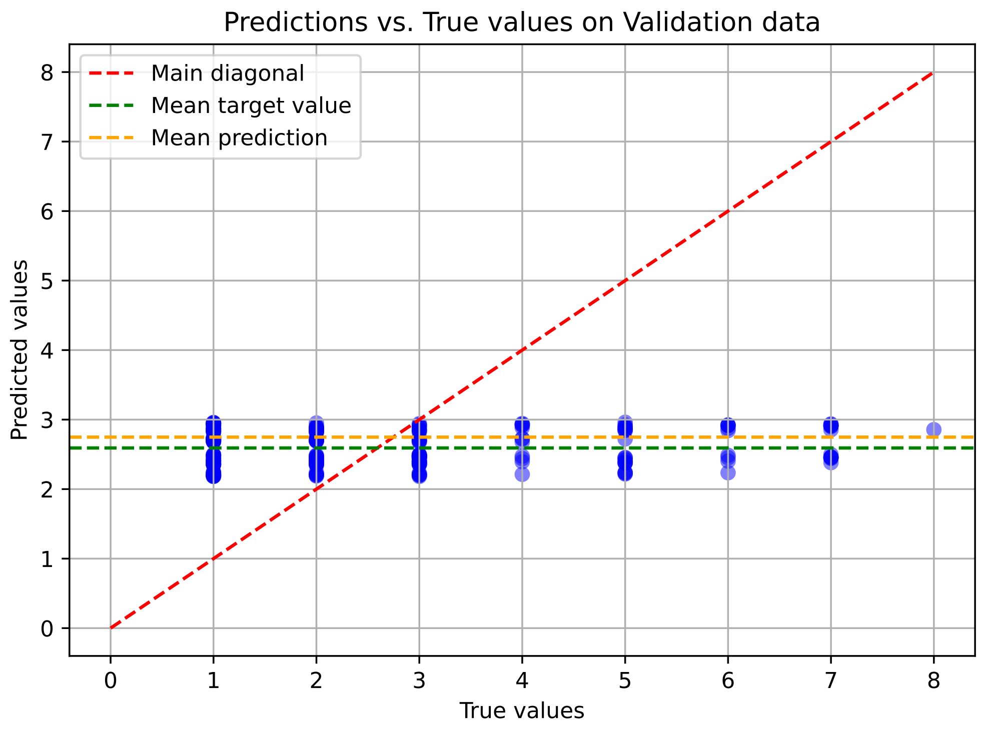


Figure 23: Predictions against ground truth for the validation dataset for the lasso model. The green dotted line is the average of the disease scores of the validation set, while the orange line is the mean of the predictions for the validation set.

Table 10: Cross-validated results for lasso Regression, the validation and training sets are evaluated using the best performing model of the CV. The CV is performed using 5 folds.

| CV results k=5 | |
|---|---|
| Average MSE | 3.83166 |
| Average $r^2$ | -0.00143 |
| Average MAD | 1.59143 |

| Training set | |
|---|---|
| MSE | 3.80245 |
| $r^2$ | 0.01781 |
| MAD | 1.58922 |

| Validation set | |
|---|---|
| MSE | 3.35168 |
| $r^2$ | 0.00233 |
| MAD | 1.46674 |

### 5.3.3 Ridge Regressor

The ridge regressor yields similar results to the previous approach, showcasing that regularisation is only marginally more effective in solving the regression problem. It similarly selects $\alpha = 0.01$ with the CV as the strength of regularisation. In Table 11 we can also appreciate that the cross-validated results are poor, having a model that essentially predicts, on average, the average of the target data. Figure 24 shows that the Ridge regression reduced less the variance of the prediction on the validation set.

Table 11: Cross-validated results for ridge regression, the validation and training sets are evaluated using the best performing model of the CV. The CV is performed using 5 folds.

| CV results k=5 | |
|---|---|
| Average MSE | 3.67137 |
| Average $r^2$ | $-0.00031$ |
| Average MAD | 1.60414 |

| Training set | |
|---|---|
| MSE | 3.87137 |
| $r^2$ | 0.00016 |
| MAD | 1.60414 |

| Validation set | |
|---|---|
| MSE | 3.36866 |
| $r^2$ | -0.00273 |
| MAD | 1.47734 |

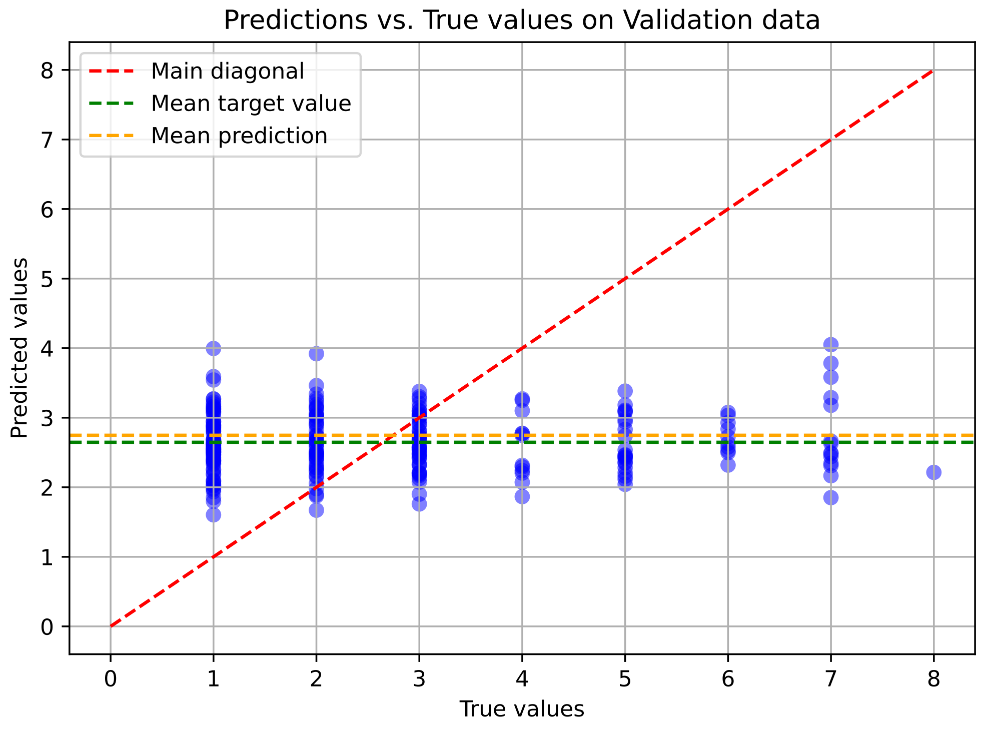


Figure 24: Predictions against ground truth for the validation dataset for the ridge model. The green dotted line is the average of the disease scores of the validation set, while the orange line is the mean of the predictions for the validation set.

### 5.3.4 Support Vector Machine

The Support Vector Machine (SVM) regressors outperform the Linear and lasso regressors, yet they encounter challenges in effectively generalising to the test set. The RBF kernel exhibited superior performance compared to the polynomial kernel and also offered significantly faster processing. Dealing with high feature counts rendered even second-order polynomial operations extremely resource-intensive. The best hyperparameters for the SVM regression model included a regularisation hyperparameter ($C$) of 0.1, and an automatically determined gamma value for the RBF kernel, contributing to enhanced model performance. The results for the best model are presented in Table 12 and Figure 25.

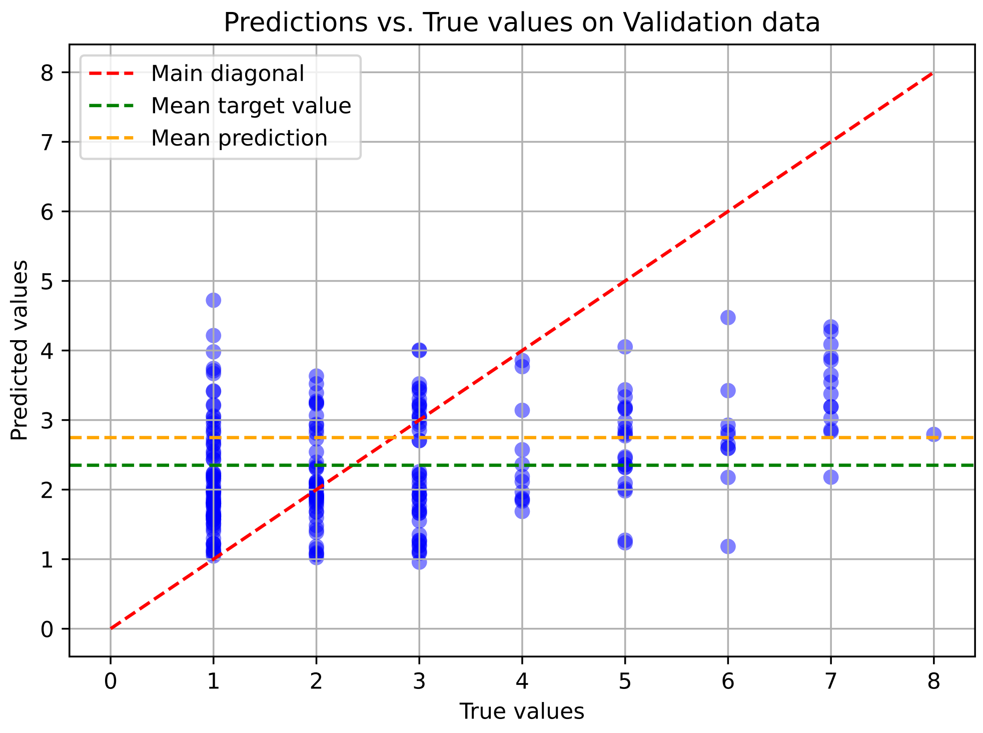


Figure 25: This figure presents the predictions given by the SVM model on the validation set. The green dotted line is the average of the disease scores of the validation set, while the orange line is the mean of the predictions for the validation set.

As in the previous cases, the predictions are close to the average value of the targets and the model struggles to predict higher disease scores.

Table 12: Metrics and CV results for the SVM regression model. The best model is selected via CV and the metrics on training and validation are computed based on it.

| CV results k=5 | |
|---|---|
| Average MSE | 3.04099 |
| Average $r^2$ | 0.20338 |
| Average MAD | 1.25889 |

| Training set | |
|---|---|
| MSE | 2.1647 |
| $r^2$ | 0.4408 |
| MAD | 0.9670 |

| Validation set | |
|---|---|
| MSE | 3.1225 |
| $r^2$ | 0.0705 |
| MAD | 1.3698 |

### 5.3.5 Random Forest Regressor

The best parameters used for the random forest model are presented in Table 13.

Table 13: Best hyperparameters for the RF regressor.

| **Hyperparameter** | **Value** |
|---|---|
| Number Estimators | 50 |
| Maximum Depth | 10 |
| Minimum Samples to Split | 5 |
| Minimum Samples per Leaf | 4 |
| Maximum Features per Node | 1 |
| Maximum Leaf per Nodes | None |
| Maximum Bootstrap Training Samples' fraction | 1 |
| Minimum Impurity Decrease | 0 |
| Bootstrap | True |

Random Forest was trained on the same data set as the last regressors and showed the same issues. However, in this case, the model was able to learn the training data quite well, as can be seen in Table 14 and Figure 27.

Table 14: Results for the RF regressor. The CV is performed using 5 folds. The validation and training sets are evaluated using the best-performing model of the CV.

| CV results k=5 | |
|---|---|
| Average MSE | 3.04099 |
| Average $r^2$ | 0.20338 |
| Average MAD | 1.25889 |

| Training set | |
|---|---|
| MSE | 1.3762 |
| $r^2$ | 0.6445 |
| MAD | 0.9085 |

| Validation set | |
|---|---|
| MSE | 3.1334 |
| $r^2$ | 0.0673 |
| MAD | 1.4285 |

The $r^2$ score on the training data suggests a strong overfitting, where the model only learns to predict the data present in the training, but then simply predicts the average for the test dataset.

In Figures 26, the graph presents the model's difficulty in accurately predicting higher values of the disease score. This phenomenon is consistent with the performance of the model on the training data, as is evident in Figure 27. In an attempt to mitigate this issue, oversampling was implemented to improve the representation of under-represented data. However, the outcomes remained comparable, indicating that an effective model should aim to discern the underlying patterns within the images, rather than rely solely on statistical representations. This element corroborated the need for a deep learning model in the next pages.

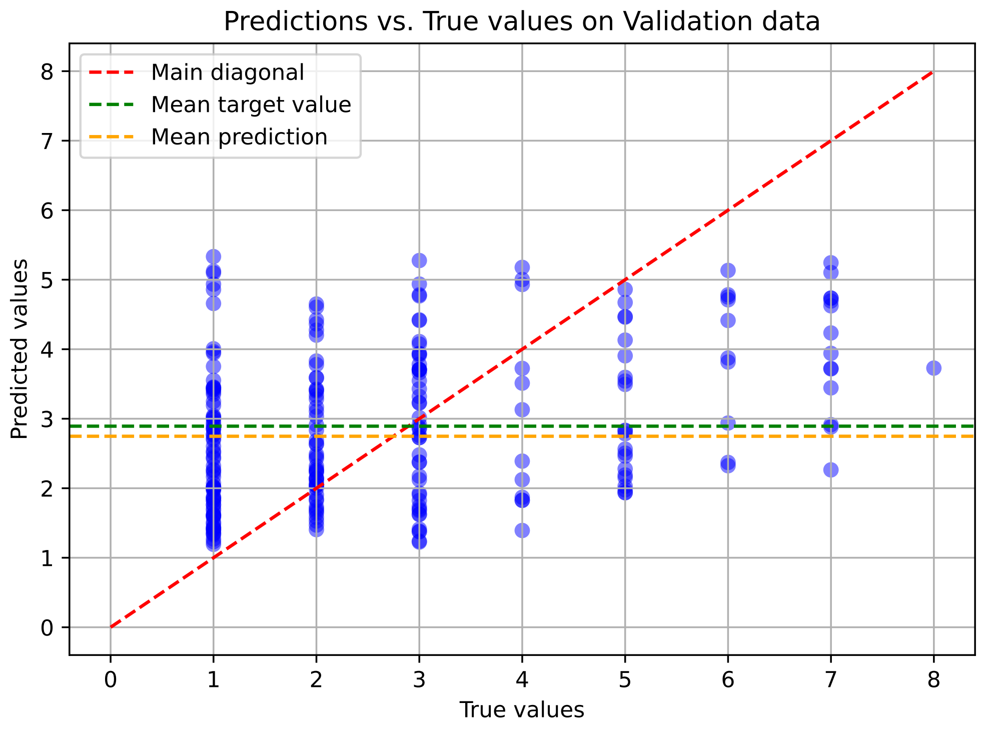


Figure 26: This figure presents the predictions given by the RF model on the validation set. The green dotted line is the average of the disease scores of the validation set, while the orange line is the mean of the predictions for the validation set.

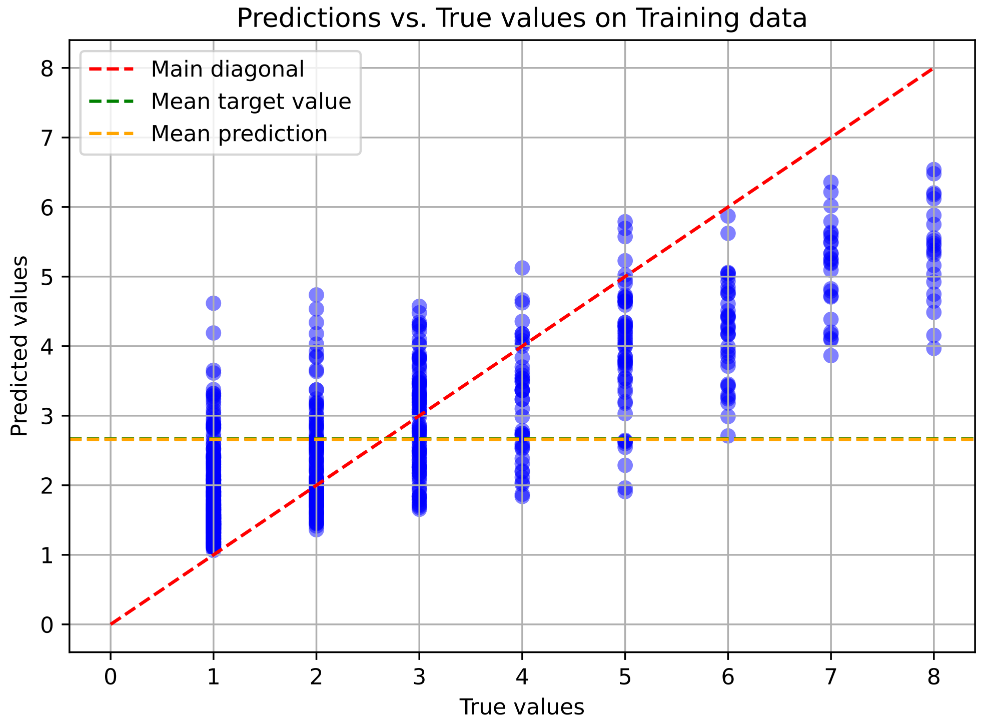


Figure 27: This figure presents the predictions given by the linear model on the training set. The green dotted line is the average of the disease scores of the train set, while the orange line is the mean of the predictions for the train set.

## 5.4 CNN-LSTM Network

This section presents the results of the simple deep learning baseline we chose. The process is described in Section 4.5.5. Since this is a deep model we did not perform cross-validation, but, on the contrary, we fine-tuned the model on our validation set. To select the best hyperparameters we applied a greedy search, based on the importance that we assigned to each hyperparameter.

### 5.4.1 Procedure of Hyperparameter Selection

Our procedure began with a deliberate decision to initialize the hyperparameters based on the default values provided by PyTorch. This initial setup served as a foundation, establishing a baseline for our subsequent investigations. Accordingly, we manually fine-tuned the initially chosen hyperparameters to ensure convergence in the model's loss. Following this, we delved into the critical hyperparameter of network depth, as discussed in Section 4.5.6, recognising its impact on the model's performance on the validation dataset. The analysis unveiled that adding layers, be them convolutional layers within the feature extractor or LSTM cells (refer to Figure 13), did not yield performance enhancements. Notably, networks with more than two convolutional layers consistently produced a negative $r^2$ score on validation, indicating a failure to capture the data variance adequately. This prompted a more in-depth investigation into the repercussions of network depth, as elucidated in Appendix B.2.1. The dependence on the depth is presented in Figure 40, 39, 41a and 41b and in Table 15 and 16.

Table 15: Model performance for different numbers of convolution layers. The metrics are computed on both the training and the validation sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| 3 conv. layers | 6.190 | 5.649 | -0.847 | -0.422 |
| 2 conv. layers | 9.858 | 10.025 | -1.942 | -1.523 |
| **1 conv. layer** | **2.124** | **1.655** | **0.366** | **0.583** |

Adding complexity to a model typically enhances validation results. However, Table 16 indicates a degradation in training set performance over time for weight decay values of 0. This phenomenon could be attributed to difficulties in reaching convergence by the network. A reason for that would be that operating with approximately 800 training image samples is often deemed insufficient (Bengio et al. 2013). In this scenario LSTMs may induce sparsity in the generated feature map, potentially leading to a sparse setup where only the model bias contributes significantly to predictions. To investigate this hypothesis, we conducted an experiment introducing increasing

Table 16: Model performance for different number of LSTM layers. The metrics are computed on both the training and the validation sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| 5 LSTM layers | 5.231 | 5.619 | -0.561 | -0.414 |
| 4 LSTM layers | 7.976 | 6.798 | -1.380 | -0.711 |
| 3 LSTM layers | 9.308 | 9.162 | -1.777 | -1.306 |
| 2 LSTM layers | 3.589 | 2.784 | -0.071 | 0.299 |
| **1 LSTM layer** | **2.124** | **1.655** | **0.366** | **0.583** |

levels of L2 weight decay (the sole source of regularisation in this architecture) to contrast that. We then assessed performance for models with 1, 2, and 3 layers. Figure 28 illustrates the results on the training set.

An observation arises by comparing Figure 28 with the results in Table 16, where a weight decay of 0 is applied. The behaviour of the three-layer network in the figure and the table present significant disparities in the performances of the $r^2$ scores on training - -1.3 compared to 0.42. Since the experimental setup remained consistent in both, it suggests potential instability in the convergence of these networks, likely stemming from the limited data available. A comprehensive investigation of this phenomenon would necessitate repeated experiments for statistical analysis. This is unfeasible to conduct given its computational burden in the scope of this thesis. Therefore, considering this as a preliminary model to assess the performances of our final model, we opted for a shallow network, selecting a stable yet superior-performing single LSTM layer network. We kept this approach also with the number of ANN Neurons, as evident in Table 18.

The next hyperparameter tested is the number of neurons in the LSTM layers. The results are presented in Table 17.

Table 17: Model performance metrics for different numbers of neurons in the LSTM layer. The metrics are computed on both the training and the validation sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| 32 LSTM neurons | 3.157 | 2.692 | 0.058 | 0.322 |
| 64 LSTM neurons | 2.132 | 1.898 | 0.364 | 0.522 |
| 128 LSTM neurons | 2.079 | 1.686 | 0.380 | 0.576 |
| 256 LSTM neurons | 2.124 | 1.655 | 0.366 | 0.583 |
| **512 LSTM neurons** | **1.620** | **1.230** | **0.517** | **0.690** |

The variant featuring an LSTM layer with 512 neurons demonstrated superior performance on the validation set. Therefore, this configuration was employed in subsequent explorations. Then, we tuned the width of the output layer. Given the poor

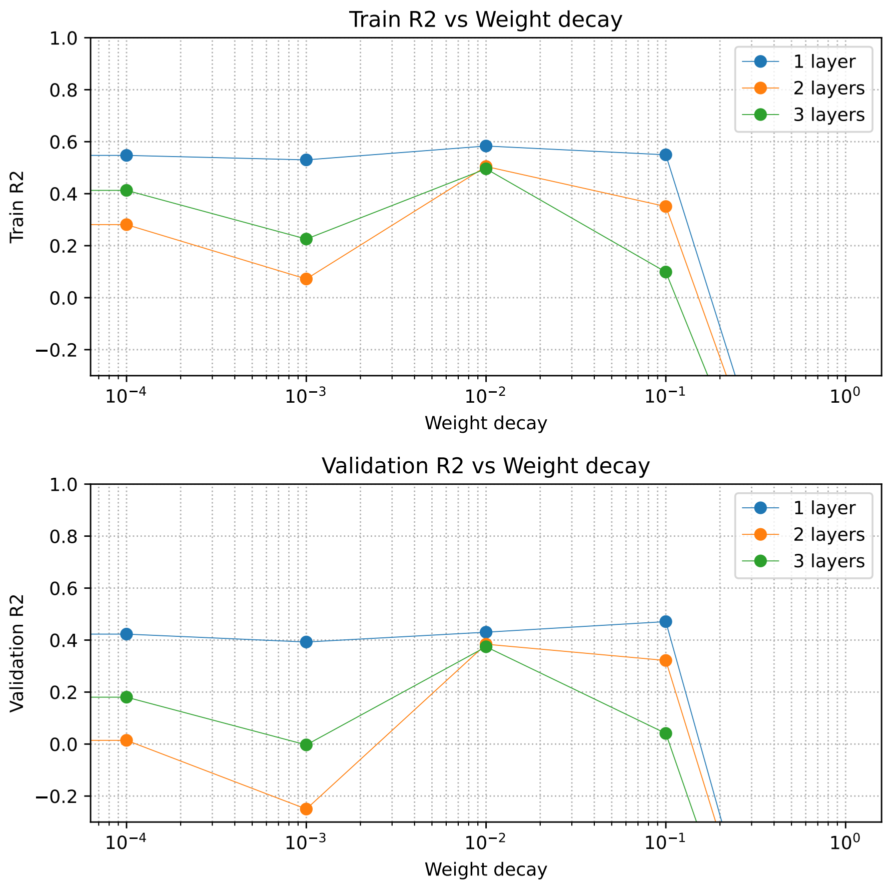


Figure 28: $r^2$ scores on the training set across varying LSTM layer depths. To enhance clarity, the y-axis is shown in the interval [-0.3, 1], all the values for weight decay equal to 1 are negative both on training and validation.

results of increasing the network depth, we always kept 1 single hidden layer in the output ANN (see Figure 13). Nevertheless, the number of neurons of the head ANN is benchmarked and the results are presented in Figure 42a and 42b; as well as in Table 18.

Table 18: Model performance for different numbers of neurons in the output ANN layer. The metrics are computed on both the training and validation sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| 32 neurons ANN | 2.024 | 1.453 | 0.396 | 0.634 |
| 64 neurons ANN | 1.743 | 1.304 | 0.480 | 0.672 |
| 128 neurons ANN | 2.009 | 1.690 | 0.401 | 0.575 |
| **256 neurons ANN** | **1.620** | **1.230** | **0.517** | **0.690** |
| 512 neurons ANN | 1.773 | 1.448 | 0.471 | 0.635 |

Regarding data augmentation, our investigation revealed that incorporating flips enhances the model's performance on the validation dataset by effectively expanding our training sample. We conducted experiments and fine-tuned various augmentation hyperparameters throughout the thesis work. We identified the most impactful ones listed below and we used them to select which combination of augmentation is optimal in this study. We also experimented with other augmentation hyperparameters, in particular different values of salt and pepper noise, but we did not report them in the thesis for brevity since they did not present good performance on validation. The results obtained are presented in Table 19.

Table 19: Metrics computed on both training and validation sets. The augmentations listed are applied on training data during training. The vertical and horizontal flips are applied in all the listed cases.

| MSE val. | MSE train | $r^2$ val. | $r^2$ train | Pepper Pr. | Salt Pr. | Gaussian Std. Dev. |
|---|---|---|---|---|---|---|
| **1.599** | **1.246** | **0.523** | **0.686** | **0** | **0** | **0.04** |
| 1.620 | 1.230 | 0.517 | 0.690 | 0 | 0 | 0.02 |
| 1.667 | 1.467 | 0.502 | 0.631 | 0.02 | 0.02 | 0.01 |
| 1.999 | 1.682 | 0.403 | 0.577 | 0.01 | 0.01 | 0.01 |
| 2.123 | 1.790 | 0.366 | 0.549 | 0.01 | 0.01 | 0.02 |
| 2.481 | 2.325 | 0.260 | 0.415 | 0.02 | 0.02 | 0.04 |
| 2.674 | 2.440 | 0.202 | 0.386 | 0.01 | 0.01 | 0.04 |

The network presents the best performance when no salt and pepper noise is applied. The Gaussian standard deviation has an impact on the validation performances, but, instead of using other hyperparameters, we decided to change the architecture itself i. Even if other alternative optimisation strategies were possible, we opted for an easy one, to meet the typical time constraints of a Master Thesis.

### 5.4.2 Best Performing Model

After the procedure as mentioned earlier, the best results correspond to the hyperparameters reported in Table 20 the best performing model. For a complete description of the hyperparameters see Table 36.

Table 20: Best hyperparameters for the custom CNN-LSTM architecture.

| **Data Augmentations** | | | | |
|---|---|---|---|---|
| Salt Pr. | Pepper Pr. | Gaussian Noise Factor | Hor. Flip | Ver. Flip |
| 0 | 0 | 0.08 | True | True |
| **Models** | | | | |
| Num. LSTM Neurons | Num CNN Filters | Kernel Size | Num Conv Layer | |
| 512 | 256 | 3 | 1 | |
| Padding | Stride | Num LSTM Layer | Num ANN Layers | |
| 1 | 1 | 1 | 1 | |
| Dropout CNN | Dropout LSTM | Dropout Output | Num. ANN Neurons | |
| 0 | 0 | 0 | 256 | |
| **Training Parameters** | | | | |
| Batch size | Learning Rate | Weight Decay ($\beta$) | Scheduler Gamma | |
| 16 | $0.0001$ | 0 | 0.85 | |
| Scheduler Step Size | Clipper Max Norm | | | |
| 50 epochs | 1 | | | |

In Figure 29, we showcase the learning curve depicting the loss function. Figure 30 illustrates the $r^2$ values on both training and validation sets throughout the training process. Additionally, the evolution of the MSE is presented in Figure 31.

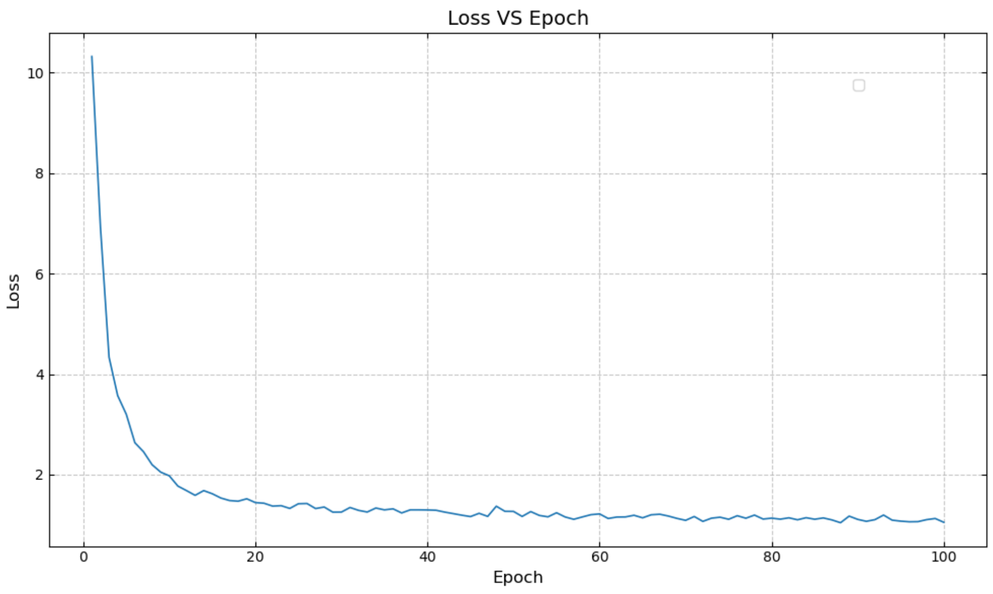


Figure 29: Loss function of the best-performing model during training.

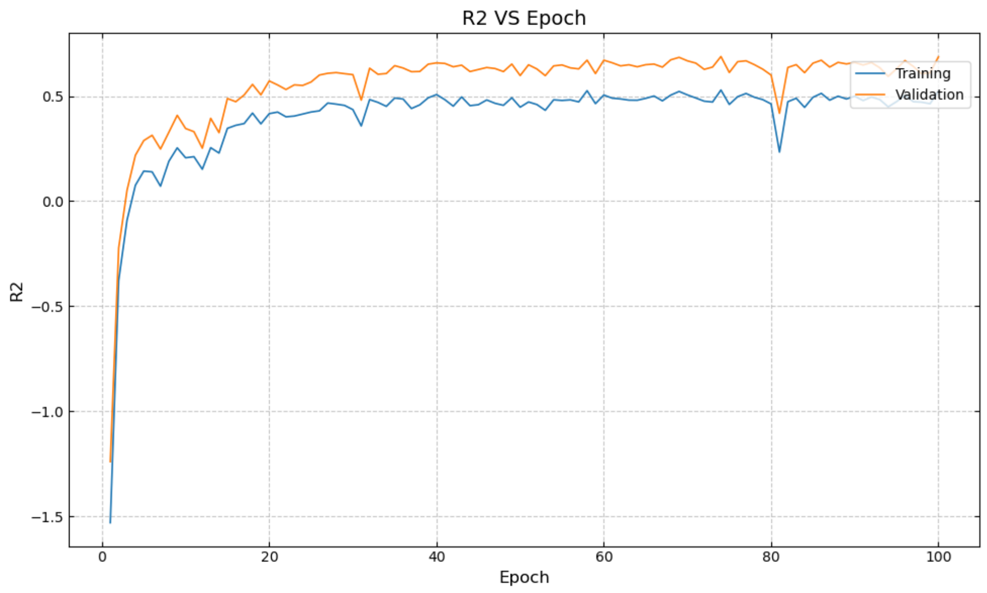


Figure 30: $r^2$ score of the best performing model.

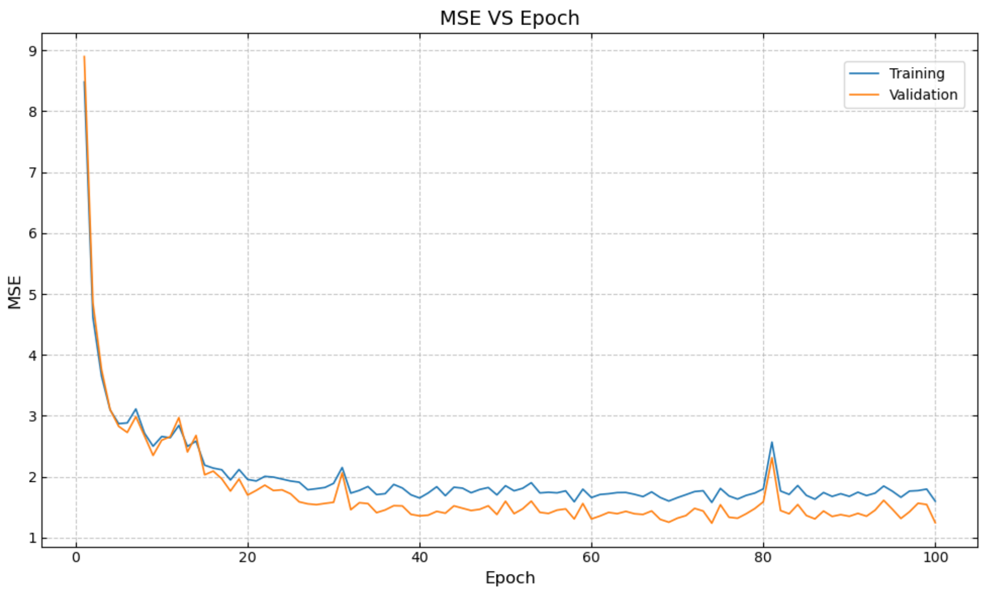


Figure 31: MSE of the best-performing model.

Those models provide a decent baseline for developing a model that uses explicitly all the time complexity and space complexity at the same time.

To conclude, the resultant metrics of the model on the three test sets defined in Section 4.3.2 are presented in Table 21.

Table 21: Evaluation results on the three test sets described in Section 4.3.2.

| | Test set 1 | Test set 2 | Test set 3 |
|---|---|---|---|
| MSE | 1.795 | 0.588 | 0.834 |
| $r^2$ | -0.084 | 0.451 | 0.268 |
| MAD | 1.054 | 0.624 | 0.423 |

### 5.4.3 Learnable Parameters

We hereby report the table of the learnable parameters for the best performing model.

Table 22: Model components and trainable parameters for the custom CNN-LSTM model.

| **Model Component** | **Parameters** |
|---|---:|
| Spatial attention weights | 8 |
| Time attention weights | 7 |
| Convolutional module | 19,200 |
| LSTM module | 1,576,960 |
| Output layer ANN | 131,585 |
| Batch normalisation after LSTM | 1,024 |
| Batch normalisation after spatial attention | 7,168 |
| Batch normalisation before spatial attention | 28,672 |
| **Total Trainable Parameters** | 1,764,624 |

## 5.5 ResNet-LSTM Network

This section presents the outcomes of the final model we introduced. As anticipated, this model outperformed all the alternatives we experimented with. Much like the preceding Section 4.5.5, we refrained from conducting cross-validation due to its impracticality.

### 5.5.1 Procedure of Hyperparameter Selection

Before attaining a functional model, for a period, we dealt with substantial bugs in both the image processing pipeline and the model's information flow. Despite these challenges, we gained valuable insights into how regularisation parameters influence the model. This understanding proved pivotal, as subsequent bug corrections resulted in a model that aligned with the performance levels detailed in Section 5.4.1 on the validation set. The initial configuration details are presented in Table 40 and Figure 46. Table 23 presents the initial performances.

Table 23: Metrics obtained before starting the procedure of hyperparameter selection. The results are reported on the validation and training sets.

| **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|
| 1.648 | 1.087 | 0.508 | 0.727 |

Following this milestone, we achieve overfitting on the training set using the new architecture, as illustrated in Figure 47 and outlined in Table 42. This procedure is obtained by running multiple experiments where we set the dropout probability to zero, and then we increase the depth of the LSTM as well as the number of neurons per layer. This outcome holds significance, indicating that our model can effectively learn the data while surpassing the performance of the Custom CNN-LSTM architecture. Table 24 presents the performances achieved when overfitting. In the subsequent experiments, we further explored the configuration space for the LSTM hyperparameters, but we did not observe substantial deviations from the first overfitting results. Hence, we retained the LSTM configurations identical to those of the overfitting model. However, we opted not to maintain the same dropout settings. Our rationale behind this decision was based on the experimental approach that, given the model's ability to overfit, reinforcing the regularisation component was crucial. Nevertheless, this adjustment increased the time needed for training the models.

Subsequently, we focused on benchmarking two critical hyperparameters: the feature representation extracted from ResNet (idx out), linked to complexity and level of detail, and the strength of regularisation, particularly for L1 feature regularisation. We opted to benchmark the former before the latter, following the sequence in Figure

Table 24: Metrics of the overfitting model. The metrics are computed both on the training and validation sets.

| **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|
| 1.483 | 0.145 | 0.558 | 0.963 |

14, after introducing dropout between the LSTM layers. Table 25 presents the benchmark between the different performances.

Table 25: Benchmark of different architectures with different depths of ResNet34 (hyperparameter "Index Out") on the validation and training sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| 3 idx out | 1.213 | 0.747 | 0.638 | 0.812 |
| **2 idx out** | **1.174** | **0.722** | **0.650** | **0.819** |
| 1 idx out | 2.348 | 1.161 | 0.299 | 0.708 |
| (1,2) idx out | 1.530 | 1.006 | 0.543 | 0.747 |
| 4 idx out | 1.221 | 0.797 | 0.636 | 0.799 |
| (2,3) idx out | 1.217 | 0.755 | 0.637 | 0.810 |

In this case, as we were expecting, the intermediate output of the ResNet feature extractor provides the best results. We then selected the value 2 for the index out hyperparameter. In Table 26 the results for the benchmark of the strength of the feature regularisation are presented.

Table 26: Benchmark of different architectures with different values for the L1 regularisation term for the lasso regressor (see Figure 14). The results are reported both on the validation and training sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| $\Omega$ : 0.01 | 1.141 | 0.838 | 0.659 | 0.789 |
| $\Omega$ : **0.001** | **1.142** | **0.693** | **0.660** | **0.826** |
| $\Omega$ : 0.00001 | 1.190 | 0.615 | 0.645 | 0.845 |
| $\Omega$ : 0 | 1.197 | 0.613 | 0.643 | 0.846 |
| $\Omega$ : 0.0001 | 1.201 | 0.626 | 0.642 | 0.842 |

Since we have several regularisation mechanisms we decided to benchmark the hypothesis where the global weight decays, following the L2 norm, to benchmark its effect. The results are presented in Table 27.

Table 27: Benchmark of different architectures with different values of weight decay. The results are reported both on the validation and training sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| $\beta$ : **0.001** | **1.094** | **0.685** | **0.673** | **0.827** |
| $\beta$ : 0 | 1.126 | 0.758 | 0.664 | 0.809 |

This observation underscores the worst generalisation achieved by eliminating weight decay. We thus decide to keep performing it.

While there was potential interest in conducting a comprehensive benchmark to assess the significance of the multiplicative constants for time ($\gamma$) and space ($\lambda$) attention, we decided against it. The decision was influenced by the fact that these attention mechanisms penalise the model based on the norm of unnormalised weights — before they are subjected to the softmax function. Given our earlier observations revealing a discernible hierarchy in the relative importance of scores, we chose to streamline our efforts and forgo the detailed benchmarking of these constants.

The next important hyperparameter we analysed is the dropout probability inside the LSTM. Table 28 presents the comparison. Notice that we did not compute benchmarked the dropout probability in the ANN output. In our design, the dropout module is not present when the ANN network has only one layer.

Table 28: Benchmark of different architectures with different values for the dropout probability inside the LSTM module (see Figure 14). The results are reported both on the validation and training sets.

| **Description** | **MSE val.** | **MSE train** | $r^2$ **val.** | $r^2$ **train** |
|---|---|---|---|---|
| **Dropout: 0** | **1.089** | **0.649** | **0.675** | **0.837** |
| Dropout: 0.1 | 1.190 | 0.662 | 0.645 | 0.833 |
| Dropout: 0.2 | 1.167 | 0.689 | 0.652 | 0.826 |
| Dropout: 0.3 | 1.156 | 0.667 | 0.655 | 0.832 |
| Dropout: 0.5 | 1.194 | 0.698 | 0.644 | 0.824 |

The key observation from our experiment is that the application of dropout didn't significantly enhance the performance on the validation set. This could be attributed to a couple of factors.

Firstly, in our model, spatial dependencies appear to have a higher significance compared to time dependencies. As a result, the ResNet feature extractor, which primarily handles spatial dependencies, does most of the heavy lifting. This reduces the impact of dropout, which is applied to the ResNet.

Secondly, we're already implementing a robust data augmentation technique by introducing salt and pepper noise with a probability of 0.01 for both salt and pepper. This technique randomly alters some features during the learning process, similar to the effect of dropout. Therefore, the additional application of dropout might have a marginally negative effect.

Lastly, since we suspect a strong impact of augmentation on the performance, we present the results of the benchmark of different augmentation techniques on our data. We decided to keep this as the last benchmark because we wanted to get a model that could generalise well with the same, reasonable, type of data augmentation. Table 29 presents the benchmark.

Table 29: Benchmark of different augmentation techniques applied on the data processed by the best-performing model obtained. We consider both the validation and training sets during the evaluation.

| MSE val. | MSE train | $r^2$ val. | $r^2$ train | Gauss. Factor | Salt Prob. | Pepper Prob. |
|---|---|---|---|---|---|---|
| **1.089** | **0.649** | **0.675** | **0.837** | **0.1** | **0.05** | **0.05** |
| 1.132 | 0.842 | 0.662 | 0.788 | 0.0 | 0.0 | 0.0 |
| 1.135 | 0.770 | 0.661 | 0.806 | 0.05 | 0.05 | 0.05 |
| 1.152 | 0.758 | 0.656 | 0.809 | 0.05 | 0.05 | 0.05 |
| 1.143 | 0.764 | 0.659 | 0.808 | 0.0 | 0.0 | 0.0 |
| 1.181 | 0.697 | 0.648 | 0.825 | 0.01 | 0.01 | 0.01 |
| 1.159 | 0.706 | 0.654 | 0.822 | 0.0 | 0.0 | 0.0 |
| 1.193 | 0.684 | 0.644 | 0.828 | 0.02 | 0.02 | 0.02 |
| 1.247 | 0.638 | 0.628 | 0.839 | 0.02 | 0.02 | 0.02 |
| 1.189 | 0.661 | 0.645 | 0.834 | 0.01 | 0.01 | 0.01 |
| 1.250 | 0.590 | 0.627 | 0.852 | 0.01 | 0.01 | 0.01 |

We conducted additional crucial experiments involving the unfreezing of the ResNet feature extractor and fine-tuning the model on multispectral data. To evaluate the impact of this modification, we compared the performance of the top-performing model with a frozen feature extractor against variations where the feature extractor was unfrozen after 25 epochs of training. This specific epoch threshold was chosen based on the observation that in previous experiments (as depicted in Figure 34), the model exhibited near-convergence around the 25th epoch. Notably, we varied the depth at which we extracted the feature map from ResNet (i.e.; the hyperparameter "Index Out") and the number of LSTM layers. Table 30 presents the results of these experiments.

In Figure 32 we present the evolution of the MSE on the training set.

Table 30: Benchmark between the best-performing model and various models that were warm started. The warm start consists of training the first 25 epochs with the ResNet34 weights frozen and then unfreezing them from the 26th epoch. Several depths of the ResNet34's feature extractor were considered as well as different numbers of LSTM layers. The metrics are reported both on the validation and training sets.

| MSE val. | MSE train | $r^2$ val. | $r^2$ train | Index =ut | Num. LSTM layers | Warmstart |
|---|---|---|---|---|---|---|
| **1.088** | **0.648** | **0.675** | **0.836** | **2** | **3** | **No** |
| 4.639 | 4.825 | -0.384 | -0.214 | 2 | 3 | Yes |
| 4.979 | 5.369 | -0.485 | -0.352 | 3 | 3 | Yes |
| 5.421 | 5.647 | -0.617 | -0.421 | 2 | 4 | Yes |
| 5.481 | 5.763 | -0.635 | -0.451 | 1 | 3 | Yes |
| 6.010 | 6.419 | -0.793 | -0.615 | 2 | 5 | Yes |
| 8.216 | 7.889 | -1.451 | -0.985 | 2 | 2 | Yes |

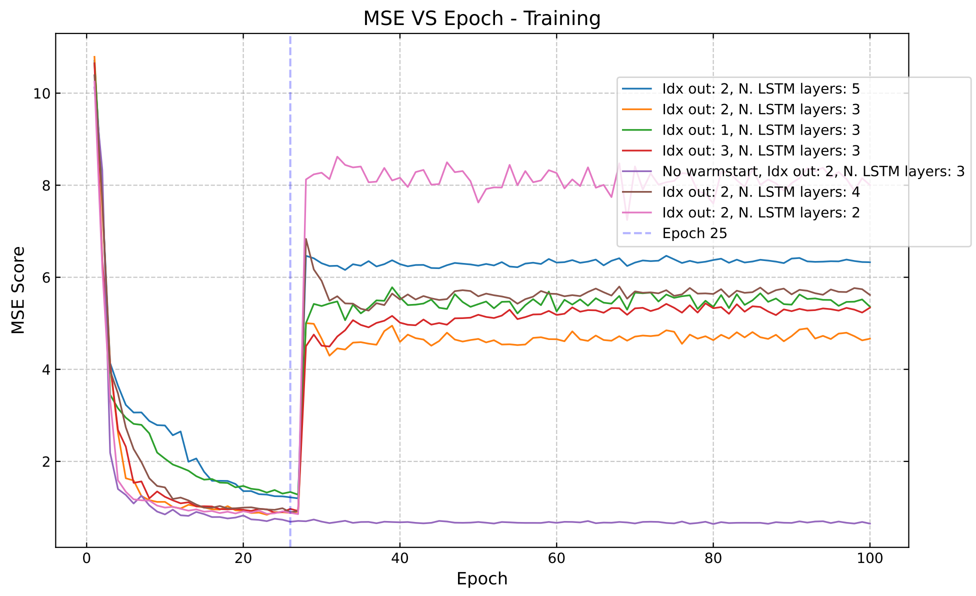


Figure 32: Benchmark between the best-performing model and some of its variations, when the weights get unlocked at the 25th epoch. Only the best-performing model previously selected has a label indicating we did not unfreeze the model. All the other models were warm-started. "Idx out" refers to the depth of the ResNet34's feature extractor.

The detrimental impact of unlocking the weights of the feature extractor on MSE is evident. This issue may stem from two factors: the disparity between the number of trainable parameters and the limited dataset, and a potential misalignment in the initialisation of batch normalisation layers. Concerning the former, the ResNet34 feature extractor boasts 21,284,672 trainable parameters, while our model, with the frozen feature extractor, has 1,873,168 trainable parameters. Consequently, unfreezing the model increases the trainable parameters by twenty-fold, surpassing the capacity of the limited dataset to adequately train the significantly deeper model. Regarding the latter, the batch normalisation within the network and ResNet adjusts to the activation maps derived from the frozen ResNet feature extractor weights. Unlocking the weights may introduce instability, leading to higher errors during training. In-depth investigations beyond the scope of this thesis are imperative to better understand and address this issue.

In summary, after choosing the hyperparameters highlighted in bold within Table 29, we arrived at the ultimate configuration for this architecture. The validation results are promising. Although superior performances could be attained, a more thorough evaluation of the hyperparameters would have not suited the time constraint of this thesis. Nevertheless, the good performances guide our research direction, establishing baseline results that pave the way for more precise mid-height disease score predictions.

### 5.5.2 Best Performing Model

After the procedure detailed in Section 5.5.1, the best results correspond to the hyperparameters reported in Table 31. For a complete description of the hyperparameters, see Table 37.

In Figure 33, we showcase the learning curve depicting the loss function. Figure 34 illustrates the $r^2$ values on both training and validation sets throughout the training process. Additionally, the evolution of the MSE is presented in Figure 35.

Table 31: Best hyperparameters for the custom ResNet-LSTM architecture.

| | | | | |
|---|---|---|---|---|
| **Data Augmentations** | | | | |
| Salt Pr. | Pepper Pr. | Gaussian Noise Factor | Hor. Flip | Ver. Flip |
| 0.01 | 0.01 | 0.01 | True | True |
| **Models** | | | | |
| Num CNN Filters | Kernel Size | Num. LSTM Neurons | | |
| 128 | 3 | 200 | | |
| Num. ANN Neurons | Num LSTM Layer | Num ANN Layers | Dropout LSTM | |
| 256 | 3 | 1 | 0 | |
| Dropout Output | | | | |
| 0 | | | | |
| **Training Parameters** | | | | |
| Batch size | Learning Rate | Weight Decay ($\beta$) | Scheduler Gamma | |
| 32 | $0.001$ | 0.001 | 0.85 | |
| Scheduler Step Size | Clipper Max Norm | | | |
| 50 epochs | 1 | | | |

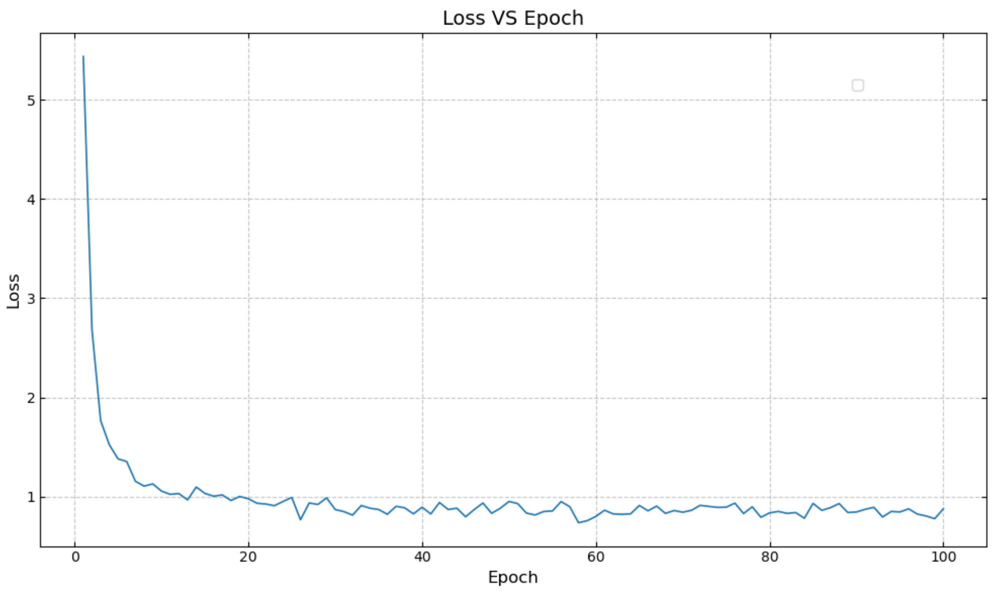


Figure 33: Loss function of the best-performing model during training.

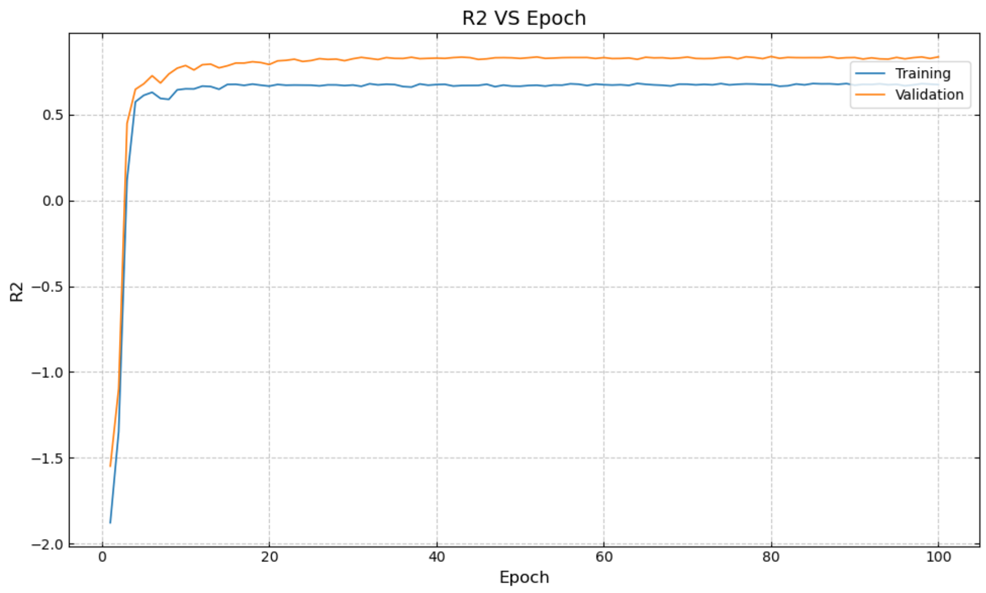


Figure 34: $r^2$ score of the best performing model.

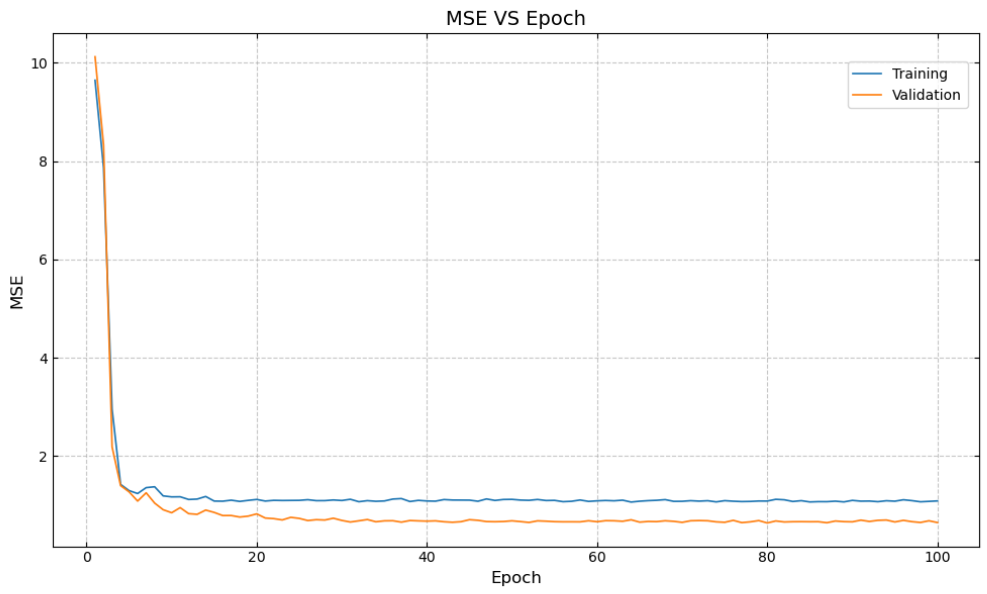


Figure 35: MSE of the best-performing model.

To conclude we present the results on the three test datasets in Table 32.

Table 32: Metrics evaluated on the 3 test sets defined in Section 4.3.2.

| | Test set 1 | Test set 2 | Test set 3 |
|---|---|---|---|
| $r^2$ | 0.181 | 0.797 | 0.620 |
| MSE | 1.564 | 0.247 | 0.380 |
| MAD | 0.960 | 0.402 | 0.364 |

We adopt a bootstrap approach to compute the confidence intervals (using a 95% coverage probability). The results are displayed in Table 33.

In Figure 36, we showcase the histograms employed in assessing the model's performance through bootstrapping. The sampling size used is 40 elements for test set 1 and 20 elements for test sets 2 and 3. The instances are drawn with replacement. This choice is the number of elements in the respective sets. The number of repetitions is $5 \cdot 10^5$.

The results show that the only test set in which we are generalising better than in the validation is test set 2 (see Section 4.3.2). The other two perform worse than the validation set. This might be closely linked to the inherent distribution of the test samples. In fact, the distribution of the target variable is unbalanced, as visible in Figure 50. In particular, the profile of the peaks in the bottom part of Figure 36

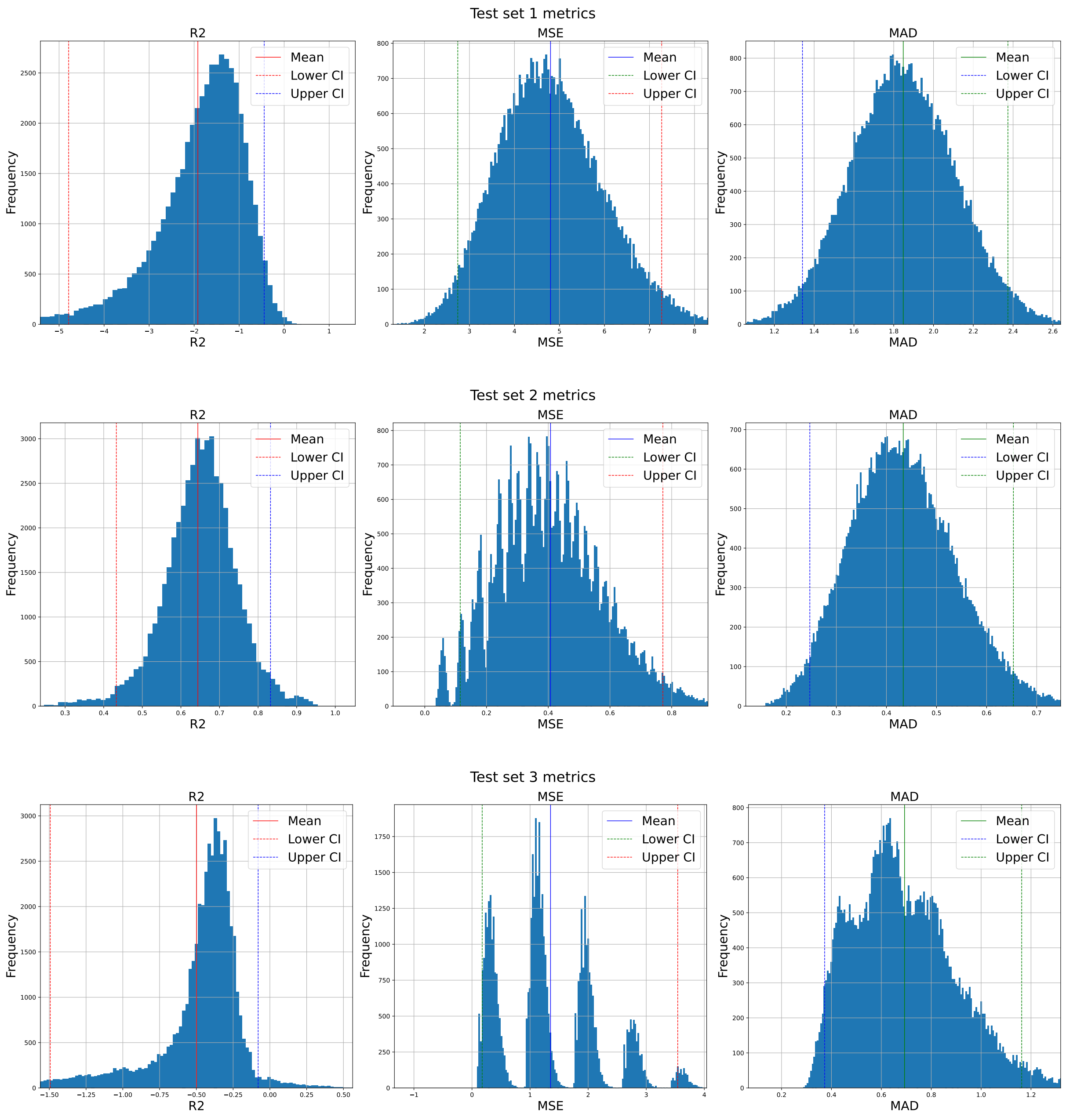


Figure 36: Bootstrap histograms representing different metrics on the three test sets defined in Section 4.3.2. The bootstrap procedure generated $5 \cdot 10^5$ different subsamples with repetitions with a sample size equal to the size of the test set considered (40 for test set 1 and 20 for both test set 2).

Table 33: Test evaluation results with confidence intervals (95% coverage probability). The bootstrap procedure generated $5 \cdot 10^5$ different subsamples with repetitions with a sample size equal to the size of the test set considered (40 for test set 1 and 20 for both test set 2).

| | Test set 1 | Test set 2 | Test set 3 |
|---|---|---|---|
| $r^2$ | 0.027 (-1.117, 0.637) | 0.712 (0.035, 0.917) | 0.516 (0.208, 0.670) |
| MSE | 1.563 (0.548, 2.886) | 0.247 (0.109, 0.416) | 0.379 (0.059, 0.980) |
| MAD | 0.960 (0.584, 1.390) | 0.402 (0.261, 0.556) | 0.364 (0.161, 0.650) |

suggests that one of the samples gets incorrectly classified with a large error. The peak structures represent the probability that the sample is drawn at random during the bootstrap procedure. This problem needs to be addressed in future iterations of this work to evaluate better and be more confident about the model's ability to generalise.

This configuration of the developed deep learning architecture shows a quantifiable performance increase compared to the Custom CNN-LSTM model (see Section 5.4). This is mainly obtained by leveraging the power of transfer learning, avoiding the problem arising by not having enough data for training.

### 5.5.3 Learnable Parameters

As a result of the experiment reported in Figure 36, the learnable parameters of the best-performing model are reported. ResNet's feature extractor was frozen, therefore it had no learnable parameters.

Table 34: Trainable parameters for the ResNet-LSTM network using the best performing model. The scheme of the architecture is reported in Figure 14.

| **Model Component** | **Parameters** |
|---|---|
| Spatial attention weights | 8 |
| Time attention weights | 7 |
| LSTM model | 1,579,008 |
| Lasso regressor | 196,608 |
| ANN head | 257 |
| Batch norm after LSTM | 512 |
| Batch norm after spatial attention | 10,752 |
| Batch norm before spatial attention | 86,016 |
| **Total Trainable Parameters** | 1,873,168 |

### 5.5.4 Model Intepretation

The ResNet-LSTM deep learning model has several regularisation mechanisms. In particular, the attention weights, and their relative importance scores, provide a built-in interpretability of the model performances. In broader terms, we could refer to it as interpretability by design in the contest of Explainable Artificial Intelligence (XAI). We propose two interpretable results: a spatial interpretation and a time interpretation.

**Spatial Interpretability**

The spatial attention scores, derived from Equation 4, were computed for 105 experiments meeting the criterion of $r^2$ scores on validation larger or equal to 0.5. Each score corresponds to one of eight plot segments detailed in Section 16. To address potential symmetries, distances of data points from the central point of the plot - 5 m from a side - were calculated. These distances were then binned, each representing an interval from the central point to an edge. Within each bin, importance values associated with each chunk were aggregated. The resulting distances were sorted in ascending order, aligning with their respective importance. This systematic approach provides insights into spatial patterns and outliers within the dataset. Figure 37 displays these results.

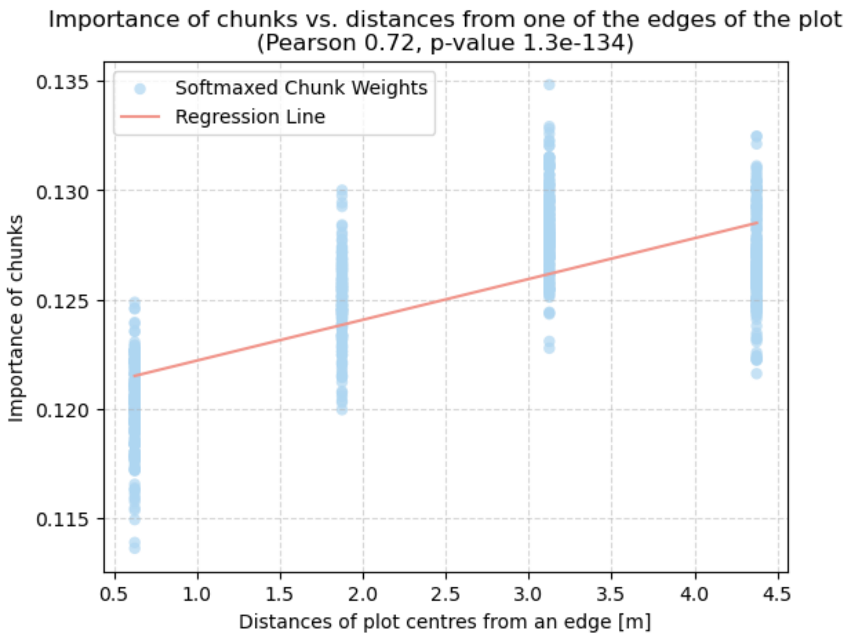


Figure 37: Position inside the experimental plot against the importance score. The models represented have an $r^2$ score on validation higher than 0.5. The position inside the plot is referred to as the centre of each chunk.

The initial key observation drawn from the plot reveals that the outer edges have a diminished impact on the model's predictive capacity compared to the central region. This trend is notably prominent within the initial 3 meters of the experimental plot. However, in the central area, the distinction is less pronounced.

**Time Interpretability**

Similarly, as in the last section, the time attention scores are reported considering the same 105 experiments. The main difference with the precedent plot is that we simply binned the time attention scores associating them to the time step they relate to. The representation of the time importance score is an insight to recognise time patterns and outlier experiments in the training process. Figure 38 displays these results.

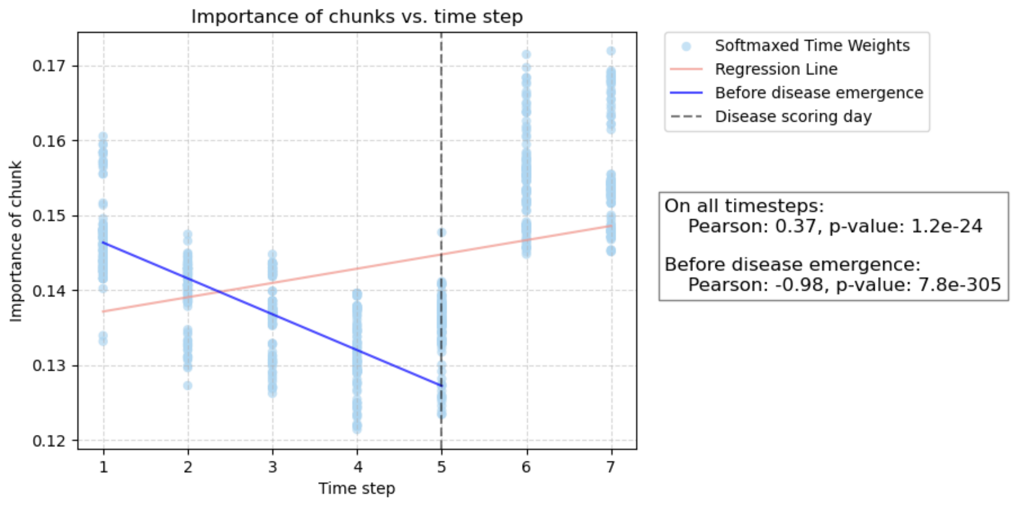


Figure 38: Time step against the importance score. The models chosen had a $r^2$ score on validation higher than 0.5. Two regression models are presented, one before the disease emergence and one considering all the time steps.

Initially, when we were designing this experiment we were expecting to measure a linear trend. Instead, we can observe that for the first time steps, noticeably the acquisitions before the "scoring day" show, over multiple independent experiments, that there is a clear downward trend for the importance scores, as evidenced by the blue line in Figure 38. The other expected, but still interesting trend is that the importance score is constant after the "scoring day". This might be since the spectral properties of the plants do not change further during the senescence phase of the plant.

In the previous regression analysis, two critical observations emerge. Firstly, it is evident that our current models lack the ability to generalize effectively. This deficiency may introduce bias into the regression analysis concerning the importance scores, as these scores are influenced by the model's inability to generalize appropriately. Secondly, it's noteworthy that the machine learning experiments are not statistically independent from one another. They share the same dataset for both training and validation, and most employ a similar hyperparameter pool due to our use of a greedy procedure for hyperparameter selection. This lack of independence is further underscored by the remarkably small p-values obtained from all regression analyses, which are less than $10^{-22}$. Despite these challenges, it's intriguing to note that all models capable of making predictions consistently assign identical weights to the importance scores.

# 6 Discussion of the Results

To address the central research question "To what extent is aerial ML-based phenotyping a viable alternative to traditional in situ phenotyping?", this thesis reports a comprehensive exploration of machine learning applications in aerial phenotyping, focused on yellow rust disease scoring. The significance of this thesis is the potential of machine learning to optimise the labour-intensive and time-consuming aspects of traditional phenotypic trait measurement.

## 6.1 Data Processing

Our methodology initially adresses the research question, "How can remote sensing image data be transformed and aggregated to extract characteristics for predicting disease scores?". The first phase involves the development of a standardised procedure for processing raw data acquired from AIT's drones. This comprehensive dataset undergoes preprocessing, including radiometric calibration, orthonormalisation, and conversion into a structured format suitable for machine learning applications (see Sections 4.2.1, 4.2.2 and 4.2.3). The application of tools such as Pix4D and QGIS streamline the extraction of geographically coherent multispectral reflectance maps, serving as the cornerstone for aerial-based phenotyping.

The introduction of Ground Control Points (GCPs) marks a significant enhancement to the entire data processing pipeline. GCPs play a pivotal role in mitigating potential biases in the data, particularly errors in geolocalisation that could lead to the misattribution of pixel groups to different plots. The inclusion of GCPs represents a substantial contribution to data quality, preventing the corruption of ground truth data. Figure 18 illustrates instances of geolocalisation bias, reaching up to 2.77 meters in the x-dimension, a critical consideration given the 1.5-meter height of the plots. The potential incorrect assignment of some pixels to a different plot might degrade data quality severely.

Subsequently, to standardize the dataset, we implement image processing measures to ensure uniform orientation for each image. This involves employing two fundamental techniques: image resampling and projective transformations. Recognising the time-intensive nature of this process and the availability of different resampling procedures, we introduce a benchmark (Figure 20) to evaluate the performance of various kernels in image reconstruction.

Despite Lanczos resampling being theoretically considered the most reliable method for image reconstruction, our practical observations reveal minimal differences compared to the bicubic filter. Notably, the runtime for bicubic is half that of Lanczos, as depicted in Figure 21. Consequently, we opt for the bicubic approach, considering its comparable effectiveness and significantly reduced processing time. Those are critical factors when contemplating scaling the procedure within a larger framework, where computing time becomes a decisive and economic consideration in this pipeline.

This approach, while present in other scientific domains such as forestry management (Miller et al. 2021), is novel in the context of airborne yellow rust disease scoring applications. The lack of application in previous studies, such as Tang et al. 2023 or Nguyen et al. 2023, can be attributed to the lower operational altitude of the drones, eliminating the need for complex image processing and geolocalisation. Our successful implementation paves the way for UAV automated disease scoring, opening new possibilities for creating standardised and reusable open-source data sets for yellow rust prediction and smart plant breeding.

## 6.2 Predictive Power at Different Altitudes

In this thesis, we address the question: "To what extent can mid-altitude UAV-acquired images effectively predict yellow rust scores, compared to low-altitude UAV and hand-held captures?". Our best predictive model shows promising performance, as outlined in Table 35.

Table 35: Synthesis of past studies on yellow rust prediction.

| Study | Key Findings | UAV | Bin. target |
|---|---|---|---|
| Moshou et al. 2004 | Demonstrated the superiority of deep learning with a Multi-Layered Perceptron (MLP) achieving 99% accuracy in single leaf detection. | No | Yes |
| Kukreja and Kumar 2021 | Utilised a Deep Convolutional Neural Network (DCNN) for wheat rust disease classification, achieving 97.16% accuracy in distinguishing healthy and diseased plants at the leaf level. | No | Yes |
| Koc et al. 2022 | Processed phenocart-acquired data to predict yellow rust infection, revealing considerable infection and exploring the influence of different time steps on prediction. | No | Yes |
| Mi et al. 2020 | Introduced a C-DenseNet architecture for wheat stripe rust disease grading, achieving a test accuracy of 97.99%. The dataset classifies leaf images with 6 levels of stripe rust infection. | No | 6 Classes |
| Tang et al. 2023 | RustNet, based on ResNet-18, achieved accuracies between 0.79 and 0.86 on low-height flights. | Yes | Yes |
| Zhang et al. 2019 | Used hyperspectral data at 30 meters altitude to forecast yellow rust presence, achieving higher accuracy (0.85) with Deep Convolutional Neural Network (DCNN) compared to basic methods. | Yes | Yes |
| Nguyen et al. 2023 | Explored a similar setup for spring wheat, emphasising the incorporation of temporal information in a 3D-CNN, achieving detection accuracy of 79% for spectral-spatio-temporal fused data. | Yes | 3 classes |

Table 35 provides a comprehensive overview of studies in yellow rust prediction, showcasing diverse approaches and their key findings. These studies employ various methodologies, from machine learning techniques to utilising different altitudes and sensors for image acquisition.

It is evident from our investigation that predicting disease scores from leaf-level images presents an inherently simpler task compared to using airborne images. The higher resolution of leaf-level images allows for the application of diverse augmentation techniques, resulting in exceptional performance in single-leaf studies (see Table 35). However, as we transition to UAV-based studies, including our own, a noticeable decline in performance becomes apparent, as corroborated by the metrics in Table 35.

In contrast, among UAV-based models, our approach, despite being marginally surpassed by Nguyen et al. 2023, stands out due to our unique deployment of the drone at substantially higher operational heights. While benchmark studies by Tang et al. 2023 and Zhang et al. 2019 outperform our model, it's important to note that they had access to considerably more data and higher-resolution images that might explain their superior performance. Moreover, our approach introduces a prediction scale from 1 to 9, deviating from the easier binary classification problem, used in the other two papers. Notably, this decision aligns more closely with field operations and presents a more meaningful approach for biologists and plant breeders.

Moving forward, the inclusion of additional data, involving an increase in the number of flights and an enhancement of ground truth disease score resolution at the sub-plot level, holds the potential for substantial improvements in our model. Our best model achieves an encouraging $r^2$ score of 0.67 on validation. Substantial work remains to be done, since the results of test sets 1 and 3, see Figure 36, exhibit the effect typical of an unbalanced target distribution, in particular for test set 3 (see Section 4.3.2). In future iterations of this work, a clear strategy to define the test sets will be the topic of the research question itself.

Despite the inherent challenges associated with our design choices, operational height and target selection that characterise our more realistic and nuanced approach, we believe that increasing the dataset size will further enhance the relevancy of our model. This expansion will allow for more robust data augmentation, synthesis, and a reduction in the generalisation error, contributing to the potential for even more meaningful results.

## 6.3 Balancing Model Simplicity and Complexity

Throughout this thesis, we embrace an iterative approach to model development. Beginning with the simplest model—a linear regressor trained on plot statistics—we progress to explore transfer learning. Guided by this approach, we addressed the research question: "How can we effectively predict phenotypic traits in crops, focusing on disease scoring, while considering the trade-off between model simplicity and complexity?".

Basic models, including regularised and non-regularised linear regression, Support Vector Machines, and Random Forests, are ineffective in predicting disease scores. The omission of time considerations in the feature design for these models results in poor performance. However, utilising basic models alongside sequential data and more nuanced features (as demonstrated in Kohonen 1990) could present an intriguing avenue for further research, particularly due to the simplicity and ease of deployment of these models. Nonetheless, we opt to delve deeper into deep learning approaches, given their prevalence in recent cutting-edge papers on yellow rust phenotyping (Nguyen et al. 2023, Shahi et al. 2023).

A significant limiting factor in our exploration into machine learning is the insufficient data at our disposal for independent network training. Transfer learning, specifically utilising ResNet feature extraction capabilities, partially mitigates this challenge by leveraging a pre-trained model. However, attempts to unfreeze the model weights reveal a decline in performance, attributed to the disparity in the number of images used during ResNet34 training and our available dataset.

This discrepancy is further influenced by the fact that ResNet is designed for RGB data, while only one of the three multispectral data cubes (see Section 4.5.6) in this thesis is RGB. The unique pixel disposition in other bandwidths potentially impacts the model's adaptability. Further investigation in the future might seek to identify whether the performance drop is attributed to non-RGB spectral cubes impacting the overall model quality.

Overall the model that we use, even though it is a deep learning one, is relatively simple, since we are using only the 3rd-level feature representation of the ResNet feature extractor (see Section 3.4.7) and a small LSTM module and a small ANN output network. The ResNet module does not have any learnable parameter, since we are keeping it frozen, while the LSTM and the ANN, together with the attention weights and the lasso feature selector total 1,873,168 learnable parameters, as detailed in Table 34.

This is still a simple model compared to deep learning models such as ResNet, with 21,797,672 learnable parameters for ResNet34. If we went deeper, as detailed in Section 38, we would start to observe degraded performances since the information would not propagate well downstream in the network. Hence, overall, we strike the balance between a relatively simple deep learning model. This is also a base to start the construction of more complex models when the 2024 field acquisition campaign is finished.

## 6.4 Designing a Domain-Specific Machine Learning Model

Our investigation concludes by addressing the intricate challenge of developing a machine learning model for aerial phenotyping that incorporates domain-specific agricultural knowledge. The introduction of an attention mechanism, inspired by neuroscience concepts, provides a framework to evaluate the significance of internal feature map elements. This addition enables the model to autonomously discern the relevance of elements within time series, plot specifics, and wavelength components, deepening our understanding of the decision-making process within the model.

The key findings are presented in Figure 37 and 38. The first figure highlights the significance of incorporating domain-specific knowledge in interpreting spatial importance scores. The higher importance scores on central plot segments in Figure 37 hold implications for understanding disease dynamics, particularly in the emergence and spread of fungal infections from the plot's centre outwards. This insight gains credibility as each plot segment undergoes independent processing by the feature extractor, hence attributing greater importance to inherently significant segments.

Another significant aspect pertains to its relevance for plant breeders. Initially, upon observing trends in Figure 37, we speculated that agronomists might exhibit bias towards monitoring the central plot sections due to the aggregation of disease measures. However, during discussions with wheat researchers from Edelhof and RWA they noted that the model's capability to consider micro-climate differences between the central and side plot sections prompted further investigation on their part.

These findings offer valuable insights that not only prompt new research directions but also suggest potential revisions in scoring practices. Thus, our study contributes meaningfully to the breeding community and advances the quest for XAI knowledge discovery in wheat breeding.

Another aspect prompting research questions is related to time attention weights. During network design, we expected that days after the disease's emergence would be most significant in model predictions. This assumption relied on the notion that the disease would alter the spectral properties of wheat leaves, with longer-lasting effects leading to changed spectral properties and higher importance scores. This trend is observable in Figure 38. However, an unexpected trend is observed: the first flight contained substantial information for the model's predictive capabilities. We posit that the information from the initial flights carries details about nuances not strictly related to plants. Consequently, L1 regularisation assigns lower importance scores to all time steps except the first in the period antecedent to the disease explosion. This inference arises from the circumstance that during those dates, the UAV could scan the ground directly without any interference from plants. This unexpected insight catalysed raising questions within Edelhof's team, prompting further investigations on their part. This impetus for research synergises effectively with the development of domain-specific models.

In conclusion, regularisation techniques offer valuable insights into model behaviour for both breeders and wheat researchers, as well as for machine learning practitioners. These techniques enable the integration of domain knowledge during model tuning and design, enhancing the overall effectiveness and applicability of the models and bridging the knowledge gap with experts.

# 7 Conclusion and Future Perspectives

In conclusion, this study explores the potential of aerial machine learning-based phenotyping as a viable alternative to traditional, labour-intensive in situ methods, with a particular focus on yellow rust disease scoring. Through a comprehensive investigation encompassing data processing methodologies, predictive power analysis at varying altitudes, and the delicate balance between model complexity and computational time, we provide valuable insights for crop monitoring and develop standardised data acquisition procedures, contributing to novel phenotyping techniques in agriculture.

Our findings demonstrate the significance of incorporating domain-specific agricultural knowledge in the design of deep learning models for aerial phenotyping. By introducing attention mechanisms inspired by neuroscience concepts, we provide a framework for evaluating the significance of internal feature map elements, thereby deepening our understanding of the decision-making process within the model.

Furthermore, our analysis reveals promising results in predicting disease scores from aerial images, particularly at mid-altitudes. While our approach may not outperform all existing models, its unique deployment at substantially higher operational heights presents a novel perspective that aligns more closely with field operations. Additionally, our decision to introduce a prediction scale from 1 to 9, rather than a binary classification approach, offers a more nuanced and meaningful approach for biologists and plant breeders.

In summary, our study contributes valuable insights to the interdisciplinary field of aerial phenotyping, bridging the gap between machine learning practitioners and agricultural experts. By leveraging domain-specific knowledge and innovative methodologies, we strive to advance the quest for knowledge discovery and facilitate meaningful applications in agricultural research and practice.

**Future Perspectives**

Looking ahead, our study lays the groundwork for future research directions in aerial phenotyping and machine learning. By addressing the inherent challenges associated with model design, data processing, and predictive power analysis, we identify areas for further exploration and refinement. These include the incorporation of additional data, as well as the exploration of more complex deep learning architectures.

While our study primarily focuses on spatial augmentations, we recognise the untapped potential of incorporating time augmentations to further enhance dataset variability and improve model performance (Wen et al. 2020). However, due to the scope of our project and resource constraints, our current emphasis has been on optimising spatial augmentations. We also intend to incorporate a spectral attention module in the Resnet-LSTM network to automatically select the channels contributing the most to the regression problems. This would enhance the model's performance and likely generate insights into the most significant bandwidths and spectral indices.

In our future endeavours, we envisage more elaborate approaches that may require a larger volume of data than currently available. To address this, we are planning a new and improved measurement campaign over the next two years. This initiative aims to acquire sub-plot level ground truth disease scores, effectively quadrupling the dataset size. Additionally, we intend to increase the frequency of drone flights to capture a more extensive range of temporal data.

Furthermore, we aim to introduce "general plant fitness" indices earlier in the plant development phase. This will enable us to generate a richer supervision signal for the model through data aggregation. Moreover, we plan to modify the model to consider various intervals between UAV acquisitions, facilitating the development of a context-aware model capable of generalising across different atmospheric conditions. This adaptation is crucial for practical applications in breeding, as it expands the model's usability beyond multispectral data acquired during specific timeframes.

An essential consideration for the upcoming experimental design is a new resampling procedure to ensure a balanced dataset. This strategic approach aims to mitigate the challenges observed in the bootstrapping procedure, as depicted in Figure 36. By establishing balanced train-test-validation splits, we aim to enhance the robustness and reliability of our model evaluations.

In summary, our future perspectives underscore our commitment to advancing aerial phenotyping methodologies through comprehensive data collection, model refinement, and methodological enhancements. These initiatives are important steps towards harnessing the full potential of machine learning in agricultural research and practice.

# Appendices

## A AGES Disease Scores

This is the official scoring proposed by AGES, it is on a scale from 1 to 9 as the scientific standard requires.

- 1: no occurrence
- 2: very low to low occurrence (only individual pustules)
- 3: low occurrence (high number of plants with low occurrence or low number of plants with average occurrence; frequent, but low number of pustules)
- 4: low to medium occurrence
- 5: medium occurrence (high number of plants with medium occurrence or low number of plants with high occurrence; all plants show pustules)
- 6: medium to high occurrence
- 7: high occurrence (all plants show medium occurrence or a high number of plants show high occurrence; high number of plants are covered in pustules)
- 8: high to very high occurrence
- 9: very high occurrence (all plants are covered in pustules)

# B Custom CNN

In this section we will present the intermediate results that we used to fine-tune and optimise the models based on the custom CNN architecture.

## B.1 Description of the Hyperparameters

Table 36: Explanation of the hyperparameters for the custom CNN_LSTM architecture.

| **Hyperparameter** | **Meaning** |
|---|---|
| pretrained_model | Indicates if pre-trained model is used (False means not using pre-trained). |
| batch_size | Number of samples processed in each batch during training. |
| dropout | Dropout rate, a regularisation technique to prevent overfitting. Used between convolutional and LSTM layers. |
| dropout_output | Dropout rate for the output layer. |
| epochs | Number of times the training dataset is passed through the neural network. |
| freeze | Indicates whether to freeze layers during training (True means freezing layers). |
| kernel_size | Size of the convolutional kernel. |
| learning_rate | Step size in each iteration toward minimising the loss function. |
| loss_function | Loss function for regression tasks. |
| num_ann_layers | Number of layers in the neural network output module. |
| num_ann_neurons | Number of neurons in neural network hidden layers. |
| num_conv_layer | Number of convolutional layers. |
| num_filters | Number of filters in convolutional layers. |
| num_LSTM_layer | Number of Long Short-Term Memory (LSTM) layers. |
| num_LSTM_neurons | Number of neurons in each LSTM layer. |
| padding | Padding added to input data before convolution. |
| stride | Step size for moving the convolutional kernel across the input. |

Table 37: Explanation of the hyperparameters for the ResNet_LSTM architecture.

| **Hyperparameter** | **Description** |
|---|---|
| batch_size | Number of samples processed in each batch during training. |
| dropout | Dropout rate between LSTM layers. |
| dropout_output | Dropout in the ANN output network |
| freeze | Indicates whether to freeze layers during training (True means freezing layers). |
| index_out | Index for the output layer. |
| L1_reg_chunks | L1 regularisation strength for chunks. |
| L1_reg_gamma | L1 regularisation strength for gamma. |
| L1_reg_timesteps | L1 regularisation strength for timesteps. |
| learning_rate | Step size in each iteration toward minimising the loss function. |
| loss_function | Loss function for regression tasks. |
| num_chunks | Number of data chunks. |
| num_LSTM_layers | Number of Long Short-Term Memory (LSTM) layers. |
| num_LSTM_neurons | Number of neurons in each LSTM layer. |
| random_initialisation_FE | Indicates if random initialisation is used for feature extraction. |
| augmenting_probability | Probability of data augmentation. |
| Gaussian_noise | Indicates if Gaussian noise is applied to the input data. |
| h_flip | Indicates if horizontal flip augmentation is applied. |
| pepper_prob | Probability of pepper noise in salt-and-pepper augmentation. |
| salt_and_pepper | Indicates if salt-and-pepper noise augmentation is applied. |
| salt_prob | Probability of salt noise in salt-and-pepper augmentation. |
| v_flip | Indicates if vertical flip augmentation is applied. |
| warmstart_epoch | Epoch at which the ResNet weight are unfrozen. |

## B.2 Graphs of Hyperparameters Selection

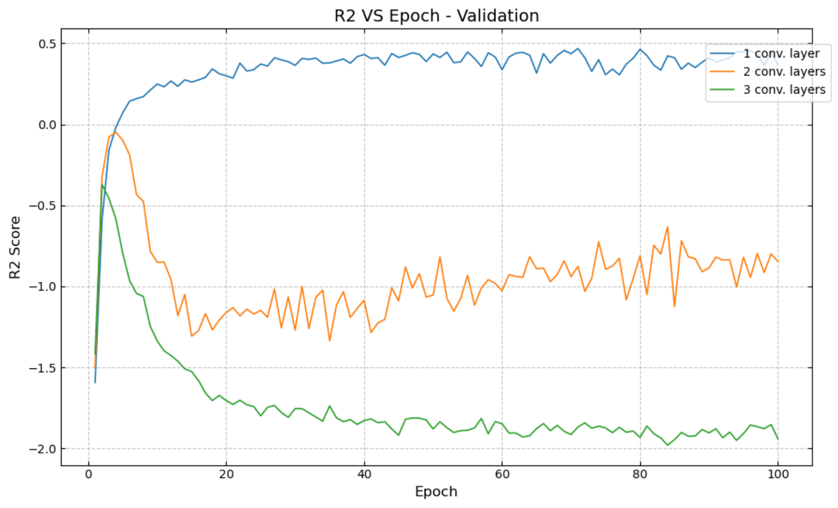


Figure 39: Benchmark between different numbers of convolutional layers.

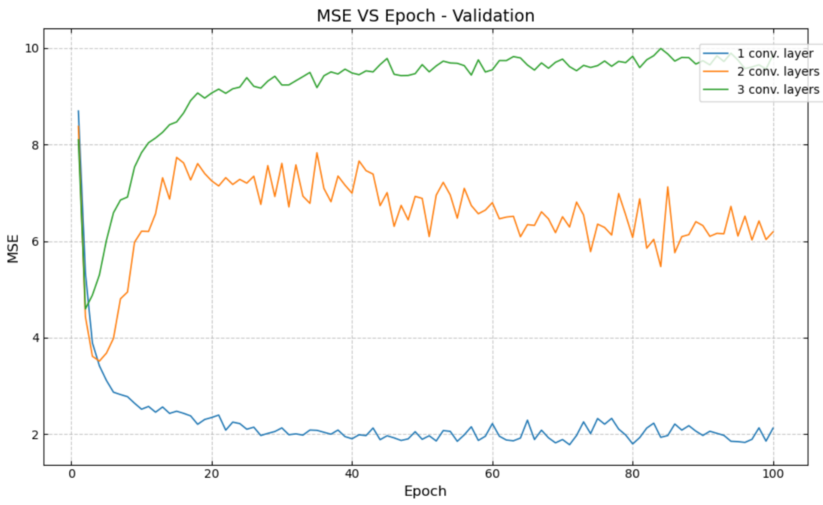
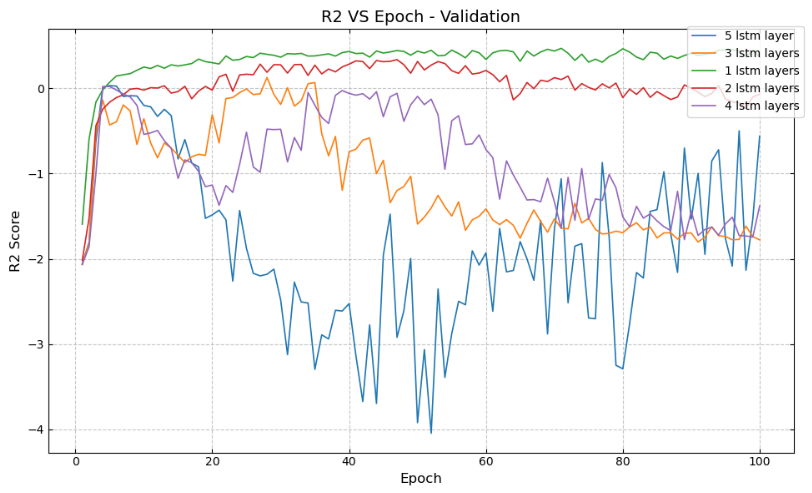
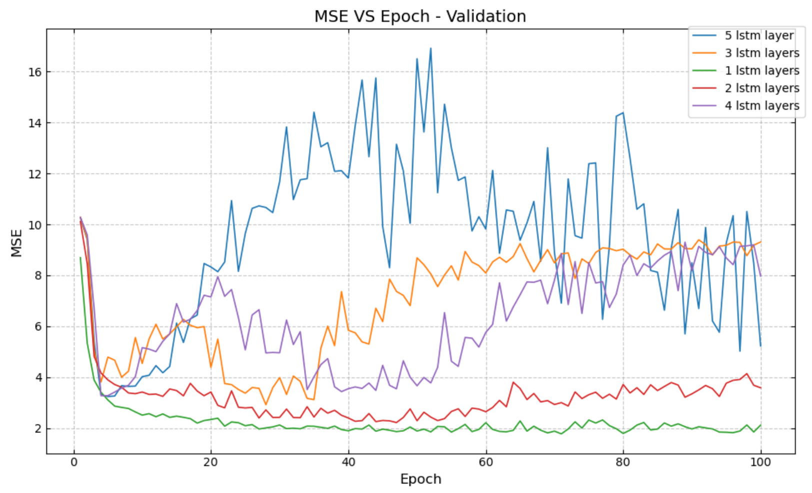


Figure 40: Benchmark between different numbers of convolutional layers.

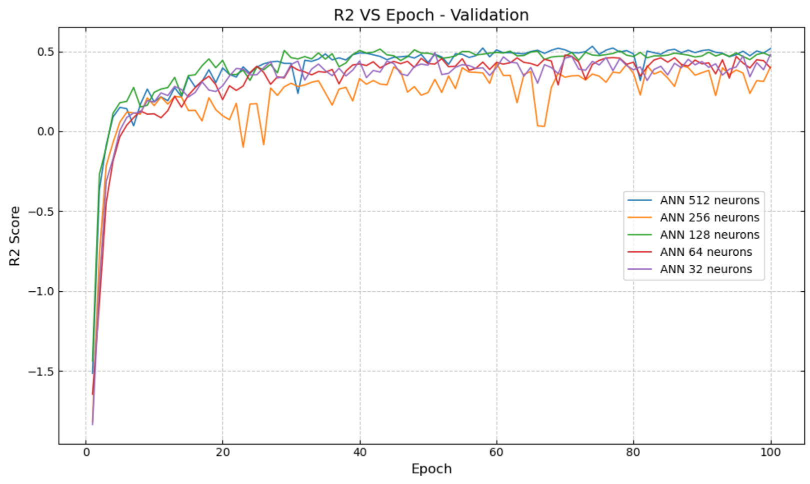


(a) Benchmark between different numbers of LSTM layers. The $r^2$ score on the validation data.

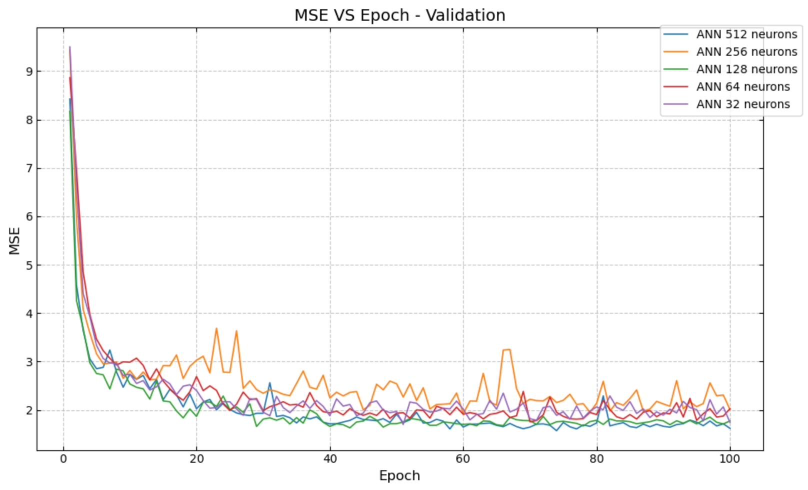


(b) Benchmark between different numbers of LSTM layers considering the MSE score on the validation data.

Figure 41: Benchmark based on the number of LSTM layers. The metrics are reported both on training and validation sets.



(a) Benchmark between different numbers of ANN neurons in the output layer. The $r^2$ score is computed on the validation set.



(b) Benchmark between different numbers of ANN neurons in the output layer. The MSE is computed on the validation set.

Figure 42: Benchmark based on the numbers of neurons in the output layer. The metrics are reported both on the validation and the training set.

### B.2.1 Deep Network Failure

In the usage of deeper networks, it became evident that the model encountered challenges in generating predictions. This was particularly emphasised by the manifestation of negative values for the $r^2$ score. To illustrate, we showcase the outcomes of the custom CNN_LSTM architecture, employing the hyperparameters specified in Table 38.

Table 38: Hyperparameters for the CNN_LSTM architecture for the custom version. The performances in this configuration are poor.

| **Hyperparameter** | **Value** |
|---|---|
| pretrained_model | False |
| batch_size | 32 |
| dropout | 0 |
| dropout_output | 0 |
| epochs | 100 |
| freeze | True |
| kernel_size | 3 |
| learning_rate | 0.001 |
| loss_function | MSE |
| num_ann_layers | 1 |
| num_ann_neurons | 256 |
| num_conv_layer | 3 |
| num_filters | 128 |
| num_LSTM_layer | 3 |
| num_LSTM_neurons | 256 |
| padding | 1 |
| stride | 1 |

To present the difficulties in generalisation of the model we will include solely the $r^2$ score, in Figure 45, and the MSE, in Figure 44, for both training and validation sets. As well as the loss function, in Figure 43.

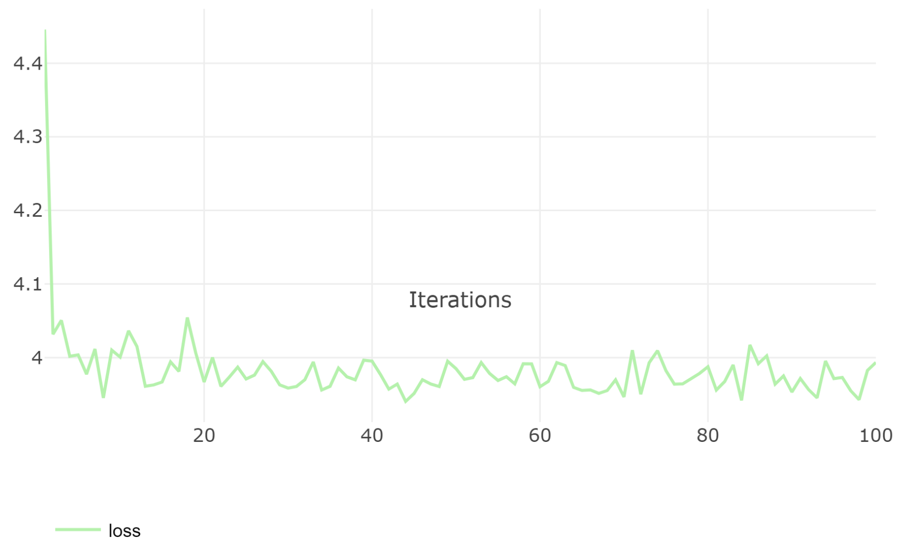


Figure 43: Loss function in the deep CNN-LSTM network. With multiple convolutional layers, the CNN-LSTM model does not converge.

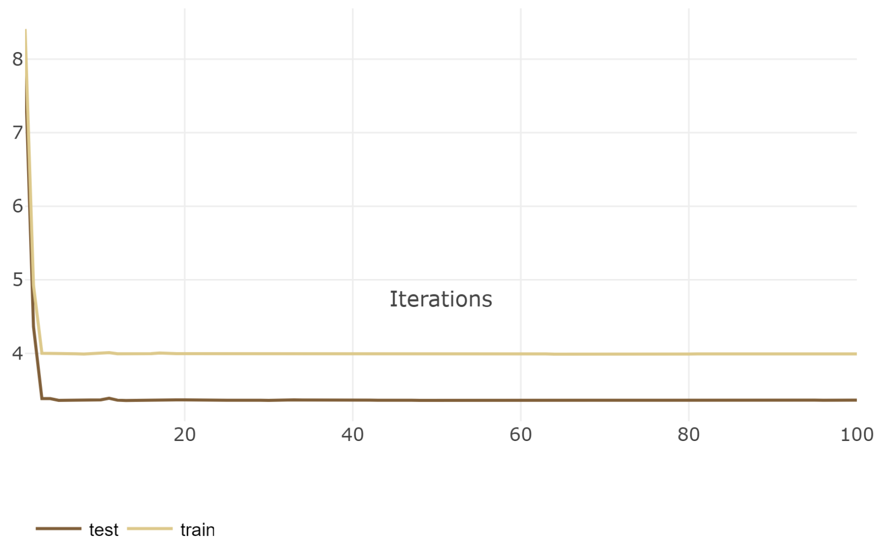


Figure 44: MSE on the training and validation sets for the CNN-LSTM model.

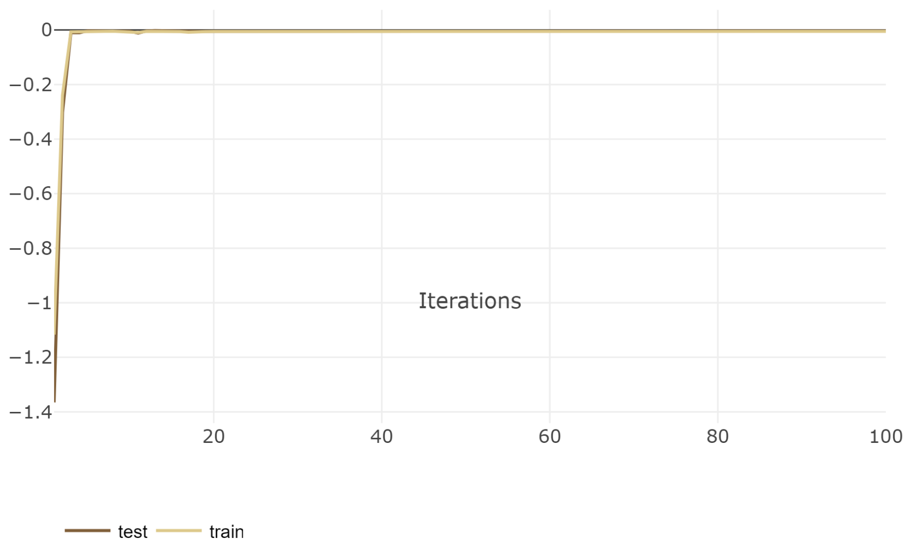


Figure 45: $r^2$ scores for training and validation sets for the CNN-LSTM model. The model does not explain any variance in the predictions.

# C ResNet-LSTM Model

In this part, we present the fine-tuning of the model.

## C.1 Initial Configuration

Table 39: Data augmentation configuration.

| **Data Augmentations** | **Value** |
|---|---|
| Salt Probability | 0.05 |
| Pepper Probability | 0.05 |
| Gaussian Noise | False |
| Horizontal Flip | True |
| Vertical Flip | True |

| **Models** | **Value** |
|---|---|
| Batch size | 64 |
| Num LSTM neurons | 200 |
| Num LSTM Layer | 3 |
| Num ANN Layers | 0 |
| Dropout Output | 0 |
| Dropout LSTM | 0.6 |
| Num ANN Neurons | 256 |
| Freeze | True |
| Index Out | 1 |
| Constant L1 for chuncks | 0.001 |
| Constant L1 for features | 0.001 |
| Constant L1 for timesteps | 0.001 |
| Learning Rate | 0.001 |
| Number Chunks | 8 |

Table 40: ResNet-LSTM initial configuration before starting the fine-tuning.

This configuration was the very first that produced results with an $r^2$ score on the validation set that was comparable with the best model in the custom CNN and LSTM set-up. Figure 46 presents the $r^2$ score on training and validation data.

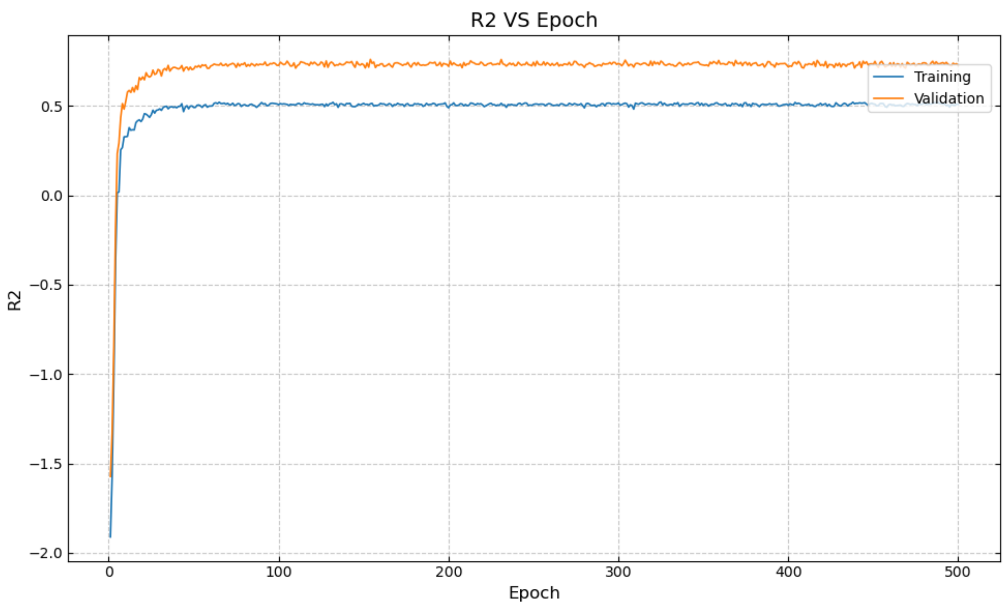


Figure 46: $r^2$ score for the first ResNet-LSTM model that obtained the same performances as the CNN-LSTM model.

## C.2 Overfit

Table 41: Data augmentations configuration.

| Data Augmentations | Value |
|---|---|
| Salt Probability | 0 |
| Pepper Probability | 0 |
| Gaussian Noise | False |
| Horizontal Flip | False |
| Vertical Flip | False |

| Models | Value |
|---|---|
| Batch size | 32 |
| Num LSTM neurons | 256 |
| Num LSTM Layer | 3 |
| Num ANN Layers | 0 |
| Dropout Output | 0 |
| Dropout LSTM | 0 |
| Num ANN Neurons | 256 |
| Freeze | True |
| Index Out | 3 |
| Constant L1 for chunks | 0.001 |
| Constant L1 for features | 0.002 |
| Constant L1 for timesteps | 0.001 |
| Num. Chunks | 8 |
| Learning Rate | 0.001 |

Table 42: ResNet-LSTM model configuration when achieving overfit.

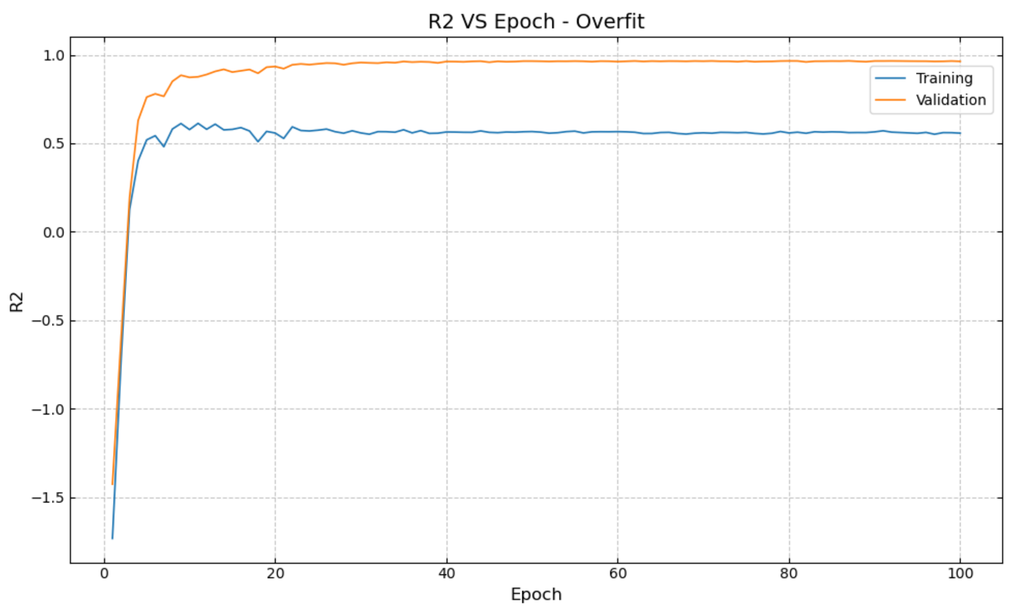
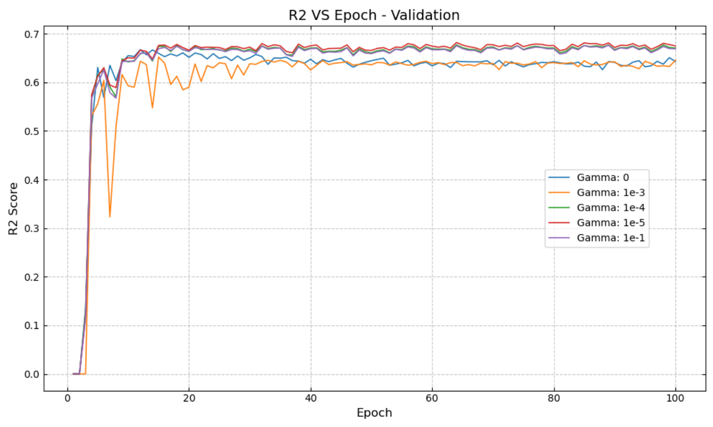
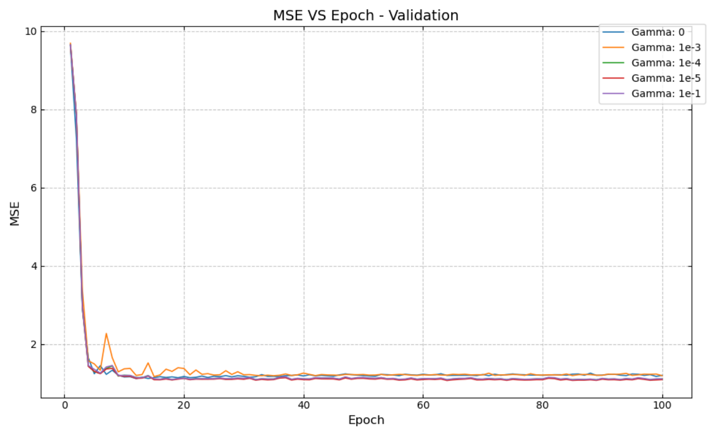


Figure 47: Overfit experiment for the ResNet-LSTM model.

## C.3 Graphs for the Selection of Hyperparameters



(a) Benchmark between different L1 regularisation factors for the lasso regressor (see Figure 14). The $r^2$ score is computed on the validation set.



(b) Benchmark between different L1 regularisation factors for the lasso regressor (see Figure 14). The MSE score is computed on the validation set.

Figure 48: Benchmark based on the strength of the L1 regularisation in the ResNet-LSTM model. The metrics are computed both on training and validation sets.

## D Disease Scores Distributions

We report the Histogram representing the distribution of the disease scores in all the data set splits reported in Section 4.3.2. Figure 49 presents the results for the training and validation sets and Figure 50 for the test sets. The specifics regarding the train, test and validation split are described in Section 4.3.2.

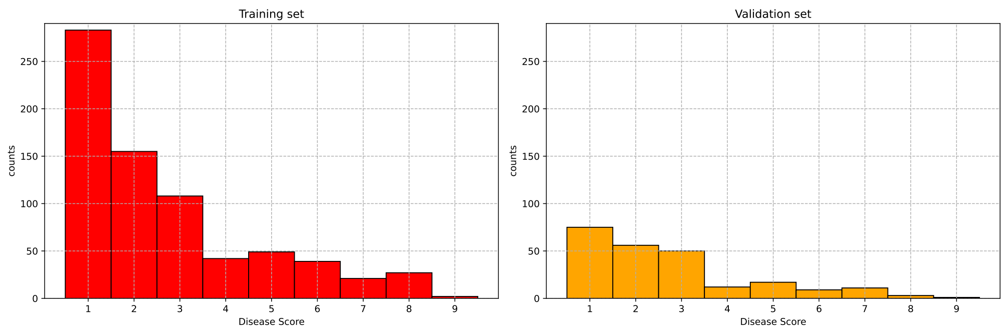


Figure 49: Distribution of the disease score in the training and in the validations sets.

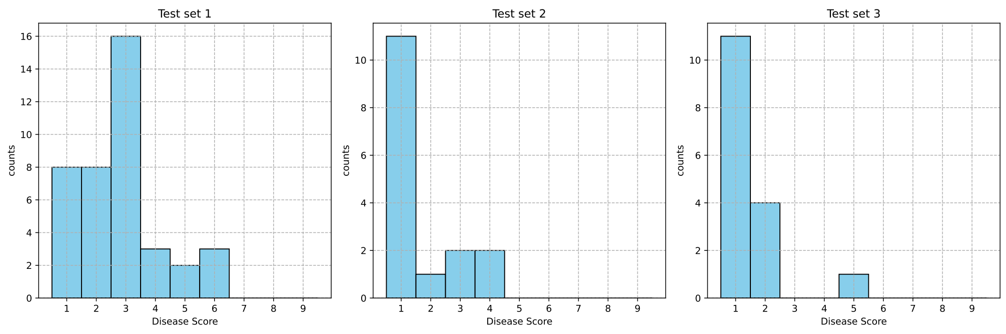


Figure 50: Distribution of the disease scores in the three test sets.

## E Pix4D Procedure

Now we will introduce the operative procedure that we defined to pass from the raw images to the orthomosaics is to first load the folder inside the Pix4d mapper.

- Open Pix4D and start a new project. Load the images, subdivided in the same folder structure as the one acquired by the camera. In our case, we used an "Altum" camera that divided the single images in spectral bands.
- In the new project window, navigate to "Advanced" settings and change the altitude of the input images to 0.
- Select ETRS (European Terrestrial Reference System) as the output coordinate system and set the output ellipsoid to 1980.
- In the processing options, uncheck "point cloud" and "mesh," leaving only the initial processing to reduce the load on your hardware.
- Proceed to the 3. DSM (Digital Surface Model) tab and choose GeoTiff as the output format. Select "Merge Tiles" since the orthomosaic is not needed for this project.
- Select the chosen indices, the 7 spectral bands and the precomputed NDVI, to have a standardised output.
- In the index calculator, ensure that it selects the area over the calibration tile. Click on "Calibration," but do not calibrate the LWIR parameter, as the camera is precalibrated.
- In the reflectance map section, click on "Merge Tiles".
- In the automatically generated quality report, verify the preview to ensure you have the desired coverage and that the point cloud is flat.
- After accessing the "Manage GCPs" panel, change the GCPs coordinate system to WGS84 (World Geodetic System 1984) and update the Geoid height according to the specifics of the acquisition system. In our project, the height is -49.54 m.
- Proceed to import GCPs. In our operations, we removed the GCPs 6 and 7 from the list, since they were deployed for preliminary experimental analyses.
- Measure the GCP in the image. Zoom out to locate the chess pattern and precisely click in the middle between 15 to 20 images. This number leads to a reconstructed mean error of less than a single pixel.

- After measuring the GCPs, go to "Process" and choose "Reoptimise".
- Now, unclick the initial processing and select the 2nd and 3rd processes. Click "Start" to initiate the processing with the selected options.

## F Kernel Trick and Mercer's Theorem

An exhaustive explanation of the kernel Trick can be found in Schölkopf 2000.

**Proposition 1** (Kernel Trick)**.** *Let $\mathcal{X}$ be the input space and $\phi : \mathcal{X} \to \mathcal{H}$ be a feature space mapping. Consider a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that satisfies Mercer's conditions. For any input vectors $x, y \in \mathcal{X}$, the dot product in the feature space $\mathcal{H}$ can be expressed using the kernel function:*

$$\langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = K(x, y)$$

*The SVM with the kernel trick implicitly operates in the feature space $\mathcal{H}$ without explicitly computing the transformation $\phi$, enabling the algorithm to efficiently handle non-linear relationships and high-dimensional feature spaces.*

**Theorem 1** (Mercer's Theorem)**.** *Let $X$ be a non-empty set and $K : X \times X \to \mathbb{R}$ be a symmetric function. $K$ is a valid kernel function, meaning there exists a mapping $\phi : X \to \mathcal{H}$ to a reproducing kernel Hilbert space $\mathcal{H}$ such that for all $x, y \in X$, the kernel function can be expressed as an inner product in $\mathcal{H}$:*

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

*Additionally, for any finite set of points $x_1, x_2, \ldots, x_n \in X$, the corresponding kernel matrix $K_{ij} = K(x_i, x_j)$ must be positive semi-definite, i.e., for any vector $c \in \mathbb{R}^n$, the inequality holds:*

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

*This condition ensures that the kernel matrix is positive semi-definite for any set of input points, guaranteeing the existence of a feature space and the validity of the kernel trick in SVMs.*

## G German Abstract

In dieser Arbeit wird die Anwendung von Techniken des maschinellen Lernens (ML) für die Phänotypisierung unter Verwendung multispektraler und multitemporaler Daten aus der Luft untersucht, wobei der Schwerpunkt auf der Identifizierung von Gelbrost bei Weizen liegt. Das Hauptziel besteht darin, die Wirksamkeit der ML-basierten Phänotypisierung aus der Luft als Alternative zu herkömmlichen in-situ-Methoden zu bewerten. Ein bahnbrechender Aspekt dieser Studie ist die Erstellung eines neuartigen Datensatzes mit multispektralen Zeitreihenbildern, die jeweils eine Versuchsparzelle in einem Weizenfeld zeigen, zusammen mit entsprechenden Gelbrost Krankheitsbewertungen durch Experten. Anschließend wird eine vergleichende Analyse zwischen verschiedenen grundlegenden ML-Modellen und Deep-Learning-Modellen zur Vorhersage von Gelbrost anhand des Datensatzes durchgeführt. Unsere Ergebnisse zeigen die Herausforderungen, denen sich ML-Basismodelle bei der genauen Vorhersage von Gelbrost gegenübersehen, im Gegensatz zu den vielversprechenden Ergebnissen eines Deep-Learning-Modells, das den Merkmalsextraktor ResNet34 verwendet. Diese Ergebnisse unterstreichen das Potenzial von ML-Ansätzen bei der Fernphänotypisierung für die Pflanzenzüchtung, insbesondere wenn Deep-Learning-Modelle mit Aufmerksamkeitsmechanismen integriert werden. Die Studie liefert wertvolle Einblicke in die Techniken der Phänotypisierung aus der Ferne, was sich auf die Verbesserung der Krankheitsüberwachung und der Anbaupraktiken auswirkt. Allerdings ist eine weitere Verfeinerung erforderlich. Die beobachtete geringere Leistung bei einem der Testsätze deutet auf einen Mangel an Daten hin, die für die Gewährleistung der Robustheit des Modells und seiner Verallgemeinerungsfähigkeit unerlässlich sind. Künftige Forschungsarbeiten könnten zusätzliche Spektralindizes und die automatische Auswahl von Spektralbändern untersuchen. Schließlich könnte das Modell mit mehr Daten auch so trainiert werden, dass es robuster gegenüber beliebigen Zeitschritten ist, was eine entscheidende Eigenschaft für seine praktische Anwendbarkeit ist.

## References

Aasen, H., E. Honkavaara, A. Lucieer, and P. J. Zarco-Tejada (2018). “Quantitative remote sensing at ultra-high resolution with UAV spectroscopy: a review of sensor technology, measurement procedures, and data correction workflows”. In: *Remote Sensing* 10.7, p. 1091.

Alvarez-Vanhard, E., T. Corpetti, and T. Houet (2021). “UAV & satellite synergies for optical remote sensing applications: A literature review”. In: *Science of remote sensing* 3, p. 100019.

Ansarifar, J., L. Wang, and S. V. Archontoulis (2021). “An interaction regression model for crop yield prediction”. In: *Scientific reports* 11.1, p. 17754.

Araus, J. L. and J. E. Cairns (2014). “Field high-throughput phenotyping: the new crop breeding frontier”. In: *Trends in plant science* 19.1, pp. 52–61.

Bai, G., Y. Ge, W. Hussain, P. S. Baenziger, and G. Graef (2016). “A multi-sensor system for high throughput field phenotyping in soybean and wheat breeding”. In: *Computers and Electronics in Agriculture* 128, pp. 181–192.

Belcher, B. T., E. H. Bower, B. Burford, M. R. Celis, A. K. Fahimipour, I. L. Guevara, K. Katija, Z. Khokhar, A. Manjunath, S. Nelson, et al. (2023). “Demystifying image-based machine learning: A practical guide to automated analysis of field imagery using modern machine learning tools”. In: *Frontiers in Marine Science* 10, p. 1157370.

Beltrame, L., J. Salzinger, P. Fanta-Jende, and C. Sulzbachner (Nov. 2024). “Practical Strategies for Automated Phenotyping: From Raw UAV Data to Multispectral Time Series for Machine Learning Applications”. In: *74th Annual Meeting 2023*. Ed. by Vereinigung der Pflanzenzüchter und Saatgutkaufleute Österreichs. University of Natural Resources and Life Sciences. Raumberg-Gumpenstein: University of Natural Resources and Life Sciences, Vienna, Austria, pp. 5–10. ISBN: 978-3-900397-13-5.

Bengio, Y., A. Courville, and P. Vincent (2013). “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828.

Berra, E. F., R. Gaulton, and S. Barr (2019). “Assessing spring phenology of a temperate woodland: A multiscale comparison of ground, unmanned aerial vehicle and Landsat satellite observations”. In: *Remote sensing of environment* 223, pp. 229–242.

Ceccarelli, T., A. Chauhan, G. Rambaldi, I. Kumar, C. Cappello, S. Janssen, and M. McCampbell (2022). “Leveraging automation and digitalization for precision agriculture: Evidence from the case studies: Background paper for The State of Food and Agriculture 2022”. In.

Chandel, N. S., Y. A. Rajwade, K. Dubey, A. K. Chandel, A. Subeesh, and M. K. Tiwari (2022). “Water stress identification of winter wheat crop with state-of-the-art AI techniques and high-resolution thermal-RGB imagery”. In: *Plants* 11.23, p. 3344.

Chang-Brahim, I., L. Koppensteiner, L. Beltrame, G. Bodner, A. Saranti, J. Salzinger, P. Fanta-Jende, C. Sulzbachner, F. Bruckmüller, F. Trognitz, M. Samad-Zamini, E. Zechner, A. Holzinger, and E. M. Molin (2023). "Reviewing the Landscape of Drought-Tolerant Winter Wheat Breeding: Integrating UAV-Based Phenotyping, GWAS, and Explainable AI for Marker-Assisted Selection". Manuscript under preparation.

Chen, W., C. Wellings, X. Chen, Z. Kang, and T. Liu (2014). "Wheat stripe (yellow) rust caused by P uccinia striiformis f. sp. tritici". In: *Molecular plant pathology* 15.5, pp. 433–446.

ClearML (2016). *ClearML*. https://clear.ml/. Accessed: Date January 2024.

Cloud, J. and K. C. Clarke (1999). "Through a shutter darkly: the tangled relationships between civilian, military and intelligence remote sensing in the early US space program". In: *Secrecy and knowledge production*, pp. 36–56.

E.S.A. (2023). *WorldView-2*. https://earth.esa.int/eogateway/missions/worldview-2. Accessed: 13-Nov-2023, ESA official website.

Fan, J., J. Zhang, S. J. Maybank, and D. Tao (2022). "Wide-angle image rectification: A survey". In: *International Journal of Computer Vision* 130.3, pp. 747–776.

Franke, J. and G. Menz (June 2007). "Multi-temporal wheat disease detection by multi-spectral remote sensing". In: *Precision Agriculture* 8 (3), pp. 161–172. ISSN: 13852256. DOI: 10.1007/s11119-007-9036-y.

Gill, T., S. K. Gill, D. K. Saini, Y. Chopra, J. P. de Koff, and K. S. Sandhu (2022). "A comprehensive review of high throughput phenotyping and machine learning for plant stress phenotyping". In: *Phenomics* 2.3, pp. 156–183.

Giménez-Gallego, J., J. D. González-Teruel, M. Jiménez-Buendía, A. B. Toledo-Moreo, F. Soto-Valles, and R. Torres-Sánchez (2019). "Segmentation of multiple tree leaves pictures with natural backgrounds using deep learning for image-based agriculture applications". In: *Applied Sciences* 10.1, p. 202.

Golubev, V. (2024). *YAECS: Yet Another Easy Configuration System*. https://github.com/valentingol/yaecs. Accessed: Date.

Gracia-Romero, A., S. C. Kefauver, and Fernandez-Gallego (2019). "UAV and ground image-based phenotyping: a proof of concept with durum wheat". In: *Remote Sensing* 11.10, p. 1244.

Haghighattalab, A., L. González Pérez, S. Mondal, D. Singh, D. Schinstock, J. Rutkoski, I. Ortiz-Monasterio, R. P. Singh, D. Goodin, and J. Poland (2016). "Application of unmanned aerial systems for high throughput phenotyping of large wheat breeding nurseries". In: *Plant Methods* 12, pp. 1–15.

Hashemi, M. (2019). "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation". In: *Journal of Big Data* 6.1, pp. 1–13.

He, K., X. Zhang, S. Ren, and J. Sun (2015). *Deep residual learning for image recognition. CoRR abs/1512.03385 (2015)*.

Hestness, J., S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou (2017). “Deep learning scaling is predictable, empirically”. In: *arXiv preprint arXiv:1712.00409*.

Hochreiter, S. and J. Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Huang, S., L. Tang, J. P. Hupy, Y. Wang, and G. Shao (2021). “A commentary review on the use of normalized difference vegetation index (NDVI) in the era of popular remote sensing”. In: *Journal of Forestry Research* 32.1, pp. 1–6.

Kennedy, P. (2003). *A Guide to Econometrics*. 5th. Cambridge: The MIT Press, pp. 205–206. ISBN: 0-262-61183-X.

Koc, A., F. Odilbekov, M. Alamrani, T. Henriksson, and A. Chawade (2022). “Predicting yellow rust in wheat breeding trials by proximal phenotyping and machine learning”. In: *Plant Methods* 18.1, p. 30.

Kohonen, T. (1990). “The self-organizing map”. In: *Proceedings of the IEEE* 78.9, pp. 1464–1480.

Kukreja, V. and D. Kumar (2021). “Automatic classification of wheat rust diseases using deep convolutional neural networks”. In: *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, pp. 1–6.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Li, S., F. Yuan, S. T. Ata-Ul-Karim, H. Zheng, T. Cheng, X. Liu, Y. Tian, Y. Zhu, W. Cao, and Q. Cao (2019). “Combining color indices and textures of UAV-based digital imagery for rice LAI estimation”. In: *Remote Sensing* 11.15, p. 1763.

Louppe, G. (2014). “Understanding random forests: From theory to practice”. In: *arXiv preprint arXiv:1407.7502*.

Lu, B., P. D. Dao, J. Liu, Y. He, and J. Shang (2020). “Recent advances of hyperspectral imaging technology and applications in agriculture”. In: *Remote Sensing* 12.16, p. 2659.

Ma, J., Y. Li, H. Liu, Y. Wu, and L. Zhang (2022). “Towards improved accuracy of UAV-based wheat ears counting: A transfer learning method of the ground-based fully convolutional network”. In: *Expert Systems with Applications* 191, p. 116226.

Martins, V. S., A. L. Kaleita, B. K. Gelder, G. W. Nagel, and D. A. Maciel (2020). “Deep neural network for complex open-water wetland mapping using high-resolution WorldView-3 and airborne LiDAR data”. In: *International Journal of Applied Earth Observation and Geoinformation* 93, p. 102215.

Mi, Z., X. Zhang, J. Su, D. Han, and B. Su (2020). “Wheat stripe rust grading by deep learning with attention mechanism and images from mobile devices”. In: *Frontiers in plant science* 11, p. 558126.

Miller, Z. M., J. Hupy, A. Chandrasekaran, G. Shao, and S. Fei (2021). “Application of postprocessing kinematic methods with UAS remote sensing in forest ecosystems”. In: *Journal of Forestry* 119.5, pp. 454–466.

Mohanty, S. P., D. P. Hughes, and M. Salathé (2016). “Using deep learning for image-based plant disease detection”. In: *Frontiers in plant science* 7, p. 1419.

Moshou, D., C. Bravo, J. West, S. Wahlen, A. McCartney, and H. Ramon (2004). “Automatic detection of ‘yellow rust’in wheat using reflectance measurements and neural networks”. In: *Computers and electronics in agriculture* 44.3, pp. 173–188.

Nex, F., C. Armenakis, c. Cramer, D. A. Cucci, M. Gerke, E. Honkavaara, A. Kukko, C. Persello, and J. Skaloud (2022). “UAV in the advent of the twenties: Where we stand and what is next”. In: *ISPRS journal of photogrammetry and remote sensing* 184, pp. 215–242.

Nguyen, C., V. Sagan, J. Skobalski, and J. I. Severo (June 2023). “Early Detection of Wheat Yellow Rust Disease and Its Impact on Terminal Yield with Multi-Spectral UAV-Imagery”. In: *Remote Sensing* 15 (13), p. 3301. ISSN: 20724292. DOI: 10.3390/rs15133301.

Oerke, E., M. Lindenthal, P. Fröhling, U. Steiner, et al. (2005). “Digital infrared thermography for the assessment of leaf pathogens”. In: *Precision Agriculture ‘05. Wageningen University Press, Wageningen, The Netherlands*, pp. 91–98.

Oerke, E.-C. (2020). “Remote Sensing of Diseases”. In: *Annual Review of Phytopathology*. DOI: 10.1146/annurev-phyto-010820. URL: https://doi.org/10.1146/annurev-phyto-010820-.

Ottoni, A. L. C., R. M. de Amorim, M. S. Novo, and D. B. Costa (2023). “Tuning of data augmentation hyperparameters in deep learning to building construction image classification with small datasets”. In: *International Journal of Machine Learning and Cybernetics* 14.1, pp. 171–186.

Ozdogan, M., Y. Yang, G. Allez, and C. Cervantes (2010). “Remote sensing of irrigated agriculture: Opportunities and challenges”. In: *Remote sensing* 2.9, pp. 2274–2304.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pierce, F. J. and P. Nowak (1999). “Aspects of precision agriculture”. In: *Advances in agronomy* 67, pp. 1–85.

Pix4D Manual (2024). *Software Manual: Table View - PIX4Dmapper*. https://support.pix4d.com/hc/en-us/articles/202557969-Software-Manual-Table-View-PIX4Dmapper.

Pix4D Support (2024). *Step 1: Before Starting a Project - 1. Designing the Image Acquisition Plan*. https://support.pix4d.com/hc/en-us/articles/202557459-Step-1-Before-Starting-a-Project-1-Designing-the-Image-Acquisition-

Plan-a-Selecting-the-Image-Acquisition-Plan-Type-PIX4Dmapper. Accessed: March 18, 2024.

Prabhakar, M., Y. Prasad, and M. N. Rao (2011). “Remote sensing of biotic stress in crop plants and its applications for pest management”. In: *Crop stress and its management: Perspectives and strategies*, pp. 517–545.

Rasmussen, J., G. Ntakos, J. Nielsen, J. Svensgaard, R. N. Poulsen, and S. Christensen (2016). “Are vegetation indices derived from consumer-grade cameras mounted on UAVs sufficiently reliable for assessing experimental plots?” In: *European Journal of Agronomy* 74, pp. 75–92.

Ray, D. K., N. D. Mueller, P. C. West, and J. A. Foley (2013). “Yield trends are insufficient to double global crop production by 2050”. In: *PloS one* 8.6, e66428.

Ray, D. K., N. Ramankutty, N. D. Mueller, P. C. West, and J. A. Foley (2012). “Recent patterns of crop yield growth and stagnation”. In: *Nature communications* 3.1, p. 1293.

Rebuffi, S.-A., S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann (2021). “Data augmentation can improve robustness”. In: *Advances in Neural Information Processing Systems* 34, pp. 29935–29948.

Saha, B., E. Koshimoto, C. C. Quach, E. F. Hogge, T. H. Strom, B. L. Hill, S. L. Vazquez, and K. Goebel (2011). “Battery health management system for electric UAVs”. In: *2011 aerospace conference*. IEEE, pp. 1–9.

Salgueiro Romero, L., J. Marcello, and V. Vilaplana (2020). “Super-resolution of sentinel-2 imagery using generative adversarial networks”. In: *Remote Sensing* 12.15, p. 2424.

Schölkopf, B. (2000). “The kernel trick for distances”. In: *Advances in neural information processing systems* 13.

Shahi, T. B., C.-Y. Xu, A. Neupane, and W. Guo (2023). “Recent advances in crop disease detection using UAV and deep learning techniques”. In: *Remote Sensing* 15.9, p. 2450.

Sharma, A., A. Jain, P. Gupta, and V. Chowdary (2020). “Machine learning applications for precision agriculture: A comprehensive review”. In: *IEEE Access* 9, pp. 4843–4873.

Shorten, C. and T. M. Khoshgoftaar (2019). “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1, pp. 1–48.

Singh, A. K., B. Ganapathysubramanian, S. Sarkar, and A. Singh (2018). “Deep learning for plant stress phenotyping: trends and future perspectives”. In: *Trends in plant science* 23.10, pp. 883–898.

Slotten, H. R. (2002). “Satellite communications, globalization, and the Cold War”. In: *Technology and Culture* 43.2, pp. 315–350.

Soufi, O. and F. Z. Belouadha (2023). “Enhancing Accessibility to High-Resolution Satellite Imagery: A Novel Deep Learning-Based Super-Resolution Approach”. In: *Journal of Environmental Treatment Techniques* 11.2, pp. 44–49.

Studley, H. and K. Weber (2011). "Comparison of image resampling techniques for satellite imagery". In: *Final Report: Assessing Post-Fire Recovery of Sagebrush-Steppe Rangelands in Southeastern Idaho* 252, pp. 185–196.

Su, J., D. Yi, B. Su, Z. Mi, C. Liu, X. Hu, X. Xu, L. Guo, and W. H. Chen (Mar. 2021). "Aerial Visual Perception in Smart Farming: Field Study of Wheat Yellow Rust Monitoring". In: *IEEE Transactions on Industrial Informatics* 17 (3), pp. 2242–2249. ISSN: 19410050. DOI: 10.1109/TII.2020.2979237.

Tang, Z., M. Wang, M. Schirrmann, K.-H. Dammer, X. Li, R. Brueggeman, S. Sankaran, A. H. Carter, M. O. Pumphrey, Y. Hu, et al. (2023). "Affordable High Throughput Field Detection of Wheat Stripe Rust Using Deep Learning with Semi-Automated Image Labeling". In: *Computers and Electronics in Agriculture* 207, p. 107709.

Tantalaki, N., S. Souravlas, and M. Roumeliotis (2019). "Data-driven decision making in precision agriculture: The rise of big data in agricultural systems". In: *Journal of Agricultural & Food Information* 20.4, pp. 344–380.

timm (2024). *Creating a model (fastai documentation)*. https://timm.fast.ai/create_model. Accessed on January 31, 2024.

Valenzuela, A., K. Reinke, and S. Jones (2022). "A new metric for the assessment of spatial resolution in satellite imagers". In: *International Journal of Applied Earth Observation and Geoinformation* 114, p. 103051.

Virnodkar, S. S., V. K. Pachghare, V. Patil, and S. K. Jha (2020). "Remote sensing and machine learning for crop water stress determination in various crops: a critical review". In: *Precision Agriculture* 21.5, pp. 1121–1155.

Volpato, L., F. Pinto, L. González-Pérez, I. G. Thompson, A. Borém, M. Reynolds, B. Gérard, G. Molero, and F. A. Rodrigues Jr (2021). "High throughput field phenotyping for plant height using UAV-based RGB imagery in wheat breeding lines: feasibility and validation". In: *Frontiers in Plant Science* 12, p. 591587.

Wang, H. and Z. Ma (2011). "Prediction of wheat stripe rust based on support vector machine". In: *2011 Seventh International Conference on Natural Computation*. Vol. 1. IEEE, pp. 378–382.

Wang, J., P. Wang, H. Tian, K. Tansey, J. Liu, and W. Quan (2023). "A deep learning framework combining CNN and GRU for improving wheat yield estimates using time series remotely sensed multi-variables". In: *Computers and Electronics in Agriculture* 206, p. 107705.

Wen, Q., L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu (2020). "Time series data augmentation for deep learning: A survey". In: *arXiv preprint arXiv:2002.12478*.

Wójtowicz, M., A. Wójtowicz, J. Piekarczyk, et al. (2016). "Application of remote sensing methods in agriculture". In: *Communications in biometry and crop science* 11.1, pp. 31–50.

Wulder, M. A., T. R. Loveland, D. P. Roy, C. J. Crawford, J. G. Masek, C. E. Woodcock, R. G. Allen, M. C. Anderson, A. S. Belward, W. B. Cohen, et al. (2019). "Current status of

Landsat program, science, and applications”. In: *Remote sensing of environment* 225, pp. 127–147.

Xie, C. and C. Yang (2020). “A review on plant high-throughput phenotyping traits using UAV-based sensors”. In: *Computers and Electronics in Agriculture* 178, p. 105731.

Yu, H., B. Kong, Q. Wang, X. Liu, and X. Liu (2020). “Hyperspectral remote sensing applications in soil: a review”. In: *Hyperspectral Remote Sensing*, pp. 269–291.

Zhang, X., L. Han, Y. Dong, Y. Shi, W. Huang, L. Han, P. González-Moreno, H. Ma, H. Ye, and T. Sobeih (2019). “A deep learning-based approach for automated yellow rust disease detection from high-resolution hyperspectral UAV images”. In: *Remote Sensing* 11.13, p. 1554.

Zhao, Y., J. Han, Y. Chen, H. Sun, J. Chen, A. Ke, Y. Han, P. Zhang, Y. Zhang, J. Zhou, et al. (2018). “Improving Generalization Based on l 1-Norm Regularization for EEG-Based Motor Imagery Classification”. In: *Frontiers in Neuroscience* 12, p. 272.

Zhou, X., Y. Kono, A. Win, T. Matsui, and T. S. Tanaka (2021a). “Predicting within-field variability in grain yield and protein content of winter wheat using UAV-based multispectral imagery and machine learning approaches”. In: *Plant Production Science* 24.2, pp. 137–151.

Zhou, Z., Y. Majeed, G. D. Naranjo, and E. M. Gambacorta (2021b). “Assessment for crop water stress with infrared thermal imagery in precision agriculture: A review and future prospects for deep learning applications”. In: *Computers and Electronics in Agriculture* 182, p. 106019.