Technical University of Applied Sciences Würzburg-Schweinfurt (THWS)
Faculty of Computer Science and Business Information Systems

Master Thesis

Sugar-beet Stress Detection using Satellite Image Time Series

Submitted to the Faculty of Computer Science and Business Information Systems at the Technical University of Applied Sciences Würzburg-Schweinfurt for completion of the degree program

Master of Science

in

Artificial Intelligence

Bhumika Laxman Sadbhave

Submitted on: August 2, 2025

Primary Supervisor: Prof. Dr. Magda Gregorová
Second Supervisor: Prof. Dr. Frank-Michael Schleif

Abstract (en)

Satellite Image Time Series (SITS) data has proven effective for agricultural tasks due to its rich spectral and temporal nature. In this study, we tackle the task of stress detection in sugar-beet fields using a fully unsupervised approach. We propose a 3D convolutional autoencoder that learns meaningful spatiotemporal representations from sequences of Sentinel-2 imagery. To enhance temporal sensitivity, we incorporate acquisition-date-specific temporal encodings that help capture the growth dynamics of sugar-beets. The learned representations are used in a downstream clustering task to separate stressed from healthy fields. Our framework is built for cross-seasonal applicability, enabling stress detection across different years without the need for extensive labeled data. We conduct comprehensive experiments to examine the effect of input variations, such as reducing spectral dimensionality and incorporating vegetation indices, to evaluate model robustness and interpretability. By developing a generalizable and scalable framework for stress detection, this study lays the foundation for future advancements in unsupervised representation learning in agriculture, with potential applications in early stress identification.

Abstract (de)

Satellitenbild Zeitreihendaten haben sich aufgrund ihrer umfangreichen spektralen und zeitlichen Eigenschaften für landwirtschaftliche Aufgaben als effektiv erwiesen. In dieser Studie befassen wir uns mit der Stresserkennung in Zuckerrübenfeldern mithilfe eines vollständig unüberwachten Ansatzes. Wir schlagen einen 3D Convolutional Autoencoder vor, der aus Sequenzen von Sentinel-2-Bildern aussagekräftige räumlich-zeitliche Darstellungen lernt. Um die zeitliche Sensitivität zu erhöhen, integrieren wir aufnahmedatumsspezifische zeitlichen Kodierungen, die helfen, die Wachstumsdynamik von Zuckerrüben zu erfassen. Die erlernten Darstellungen werden in einem nachgelagerten Clustering-Prozess verwendet, um gestresste von gesunden Feldern zu unterscheiden. Unser Framework ist saisonübergreifend einsetzbar und ermöglicht die Stresserkennung über verschiedene Jahre hinweg, ohne dass umfangreiche gelabelte Daten erforderlich sind. Wir führen umfassende Experimente durch, um die Auswirkungen von Eingangsvariationen zu untersuchen, wie z. B. die Reduzierung der spektralen Dimensionalität und die Einbeziehung von Vegetationsindizes, um die Robustheit und Interpretierbarkeit des Modells zu bewerten. Durch die Entwicklung einer verallgemeinerbaren und skalierbaren Rahmens zur Stresserkennung legt diese Studie den Grundstein für zukünftige Fortschritte im unüberwachten Repräsentationslernen in der Landwirtschaft, mit potenziellen Anwendungen in der rechtzeitigen Stressidentifizierung.

Acknowledgment

I would like to sincerely acknowledge the support and guidance I received throughout the thesis. Firstly, I would like to express my deep gratitude to my supervisor, Prof. Magda Gregorová, for giving me the opportunity to explore the field of representation learning and satellite imaging. Her guidance, patience, and insightful feedback have been instrumental in shaping both this work and my personal growth. I sincerely thank Philip Väth for valuable technical insights throughout the thesis.

A heartfelt thanks to Gunther Schorcht and Denise Dejon for warmly welcoming me into the Greenspin office. Their continued support made this experience truly enjoyable and enriching, and inspired me to pursue satellite imaging as a future career path.

I am deeply grateful to my family and friends for their unwavering support throughout my studies. A special thanks to my mother, whose enduring motivation and trust empowered me, and to Dibyanshu, for the encouragement and grounding support throughout my journey.

Contents

Notations

x, y	Scalars
$\mathbf{x}$, $\mathbf{y}$	Vectors
A, B	Matrices, raster images
$A^\top$	Transpose of matrix A
a_{ij}	(i, j)-th element of matrix A
$t1 - t7$	Temporal instances from 1 to 7
T	Total number of temporal instances
C	Total number of channels
P	Patch label
α	Sub-patch-to-patch threshold
$\cdot$	Element-wise or scalar multiplication
$\mathcal{L}$	Loss function
$\in$	Belongs to to a set
$\sum(\cdot)$	Summation of elements
$\|\| \cdot \|\|$	Euclidean norm (L2 norm)
$\lceil . \rceil$	Ceil of a value
$\| \cdot \|$	Cardinality of a set (number of elements)
$\arg\min$	Argument that minimizes a given function
μ	Centroid of a cluster
$D(i, j)$	Dissimilarity or distance between data points i and j
d	Dimension of the latent variable in autoenencoders
$X \in \mathbb{R}^{H \times W \times C \times T}$	4D tensor with height H, width W, channels C and timesteps T
Sentinel-2 image	Raw Sentinel-2 satellite image
Sugar-beet field patch	Sugar-beet fields extracted from Sentinel-2 images
Sub-patch	Sugar-beet patches further divided into smaller sub-patches
Strategy A, B	Temporal Encoding strategies
B10	Tensor composed of 10 Sentinel-2 bands
MVI	Tensor composed of Multiple Vegetation Indices
B4	Tensor composed of 4 Sentinel-2 bands

Acronynms

SITS	Satellite Image Time Series
EU	European Union
SBR	Sugar Beet Rust
CLS	Cercospora Leaf Spot
RGB	Red-Green-Blue
NIR	Near Infrared
SWIR	Shortwave Infrared
VI	Vegetation Index
NDVI	Normalized Difference Vegetation Index
EVI	Enhanced Vegetaion Index
MSI	Moisture Stress Index
PWD	Pine Wilt Diseases
UAV	Unmanned Aerial Vehicle
NASA	National Aeronautics and Space Administration
ISRO	Indian Space Research Organization
RISAT-I	Radar Imaging Satellite for All-weather earth observation
SVM	Support Vector Machine
PCA	Principal Component Analysis
MLP	Multi-layer Perceptron
CNN	Convolutional Neural Network
R-CNN	Region based Convolutional Neural Network
VGG	Visual Geometry Group
YOLOv4	You Only Look Once version 4
ViT	Vision Transformer
TSViT	Temporo-Spatial Vision Transformer
AE	Autoencoder
MAE	Masked Autoencoder
SatMAE	Masked Autoencoder for Satellite data
MSE	Mean Squared Error
BCE	Binary Cross-Entropy

1 Introduction

Often referred to as 'Königin der Feldfrüchte' (queen of field crops) [52], sugar beet plays a vital role in Germany, serving as a primary source for sugar production as well as for other industrial uses. However, the cultivation of sugar-beets faces increasing challenges due to various stress factors such as adverse environmental conditions, diseases, and pests. These stressors not only reduce crop yield but also cause significant economic losses for farmers and the agricultural sector as a whole. Accurate detection of stress in sugar-beet fields is therefore essential to enable proper interventions, minimize crop loss, and improve overall productivity.

Traditionally, plant health monitoring and disease detection have relied on manual inspection or analysis of conventional RGB images [36, 26]. While useful, these methods are often labor-intensive and limited in scale. The advent of satellite remote sensing has revolutionized agricultural monitoring by providing access to multispectral imagery captured over time, enabling the analysis of temporal changes in crop health [4]. This multispectral and temporal data allows for more detailed and dynamic assessments of vegetation health, stress, and land cover characteristics across large spatial scales.

Most existing research leveraging multispectral satellite data for agricultural applications primarily focuses on supervised learning techniques, which require extensive labeled datasets for training models to classify crops or detect stress [68]. However, labeled data is often scarce or expensive to obtain, especially for large-scale or diverse agricultural landscapes. To address this, unsupervised learning methods have gained attention as they do not rely on labeled data for model training. More recently, advances in representation learning and self-supervised learning have demonstrated the ability to pre-train models without labels, which can then be fine-tuned on downstream supervised tasks using limited labeled data [28, 70, 57].

In this thesis, we explore stress detection in sugar-beet fields using a fully unsupervised learning framework. Specifically, we employ representation learning for feature extraction from Satellite Image Time Series (SITS) data, followed by clustering as a downstream task to distinguish stressed from healthy crops. Due to limited availability of labeled data, ground truth labels are reserved solely for evaluation purposes.

The thesis begins by clearly defining the problem statement, outlining the objectives and key challenges, and providing an overview of the proposed stress detection system.

This is followed by background information on sugar-beets and multispectral imaging, and then a review of relevant literature and previous studies in the domain. Through detailed data preprocessing and modeling workflows, we address the challenges and achieve the stated objectives. Additionally, a series of experiments assess how varying data configurations affect the model performance. As part of this work, we also examine an exploratory approach using masked autoencoders for satellite imagery (SatMAE) [10]. Although this method did not produce satisfactory results, it provided valuable insights into the potential of this architecture for unsupervised stress detection.

Through this thesis, we establish a solid foundation for a scalable and generalizable stress detection system for sugar-beets. While the proposed approach operates in a fully unsupervised manner due to limited labeled data, the system is designed to be refined and enhanced as more labeled data becomes available. This work paves the way for future research and practical improvements, allowing the existing framework to be extended towards diverse objectives and the development of more reliable, timely, and actionable stress detection solutions.

1.1 Problem statement

Sugar beet fields are extensive and geographically distributed, making manual inspection for stress detection labor-intensive, time-consuming, and infeasible at scale. Furthermore, access to high-resolution images and annotated data is limited and often costly for small-scale industries in the agriculture sector. This study proposes an unsupervised approach for stress detection in sugar-beet fields using publicly available Sentinel-2 satellite data. By leveraging both spectral and temporal information from the multispectral images across the full crop growth cycle—from early leaf development to harvest—we propose to develop a low-cost, scalable stress detection system that operates without the need for extensive labeled training data.

The focus of this work is on identifying general stress patterns in sugar-beet fields, including those caused by disease. However, the precise identification of specific stress types or causal agents is beyond the scope of this study.

1.2 Objectives and challenges

The primary objective of this study is to develop an unsupervised framework for detecting stress in sugar-beet fields using publicly available Sentinel-2 satellite imagery. By leveraging both the spectral richness and temporal dynamics of the multispectral data,

the framework enables generalization to data from different growth seasons and years. A key focus is to ensure that all stressed fields are correctly identified (as detecting stress is critical), while also minimizing the incorrect categorization of healthy fields as stressed.

An additional objective of this study is the generation of localized stress maps. These maps highlight specific stress-affected regions within individual sugar-beet fields, potentially reducing the need for exhaustive manual inspections.

The study addresses the following challenges:

- **Limited availability of labeled data:** Only about 5% of the dataset includes labeled sugar-beet fields, restricting the use of supervised methods and requiring alternative strategies.
- **High variability in field sizes:** Field areas range from 0.14 to 52.44 hectares, making it impractical to use a fixed-size input without introducing bias or information loss.
- **Temporal inconsistencies:** Cloud cover often disrupts the temporal continuity of Sentinel-2 imagery, complicating the coverage and modeling of the complete crop growth cycle.
- **Coarse ground truth labels:** Available labels are provided at the field level, while stress may occur only in localized regions. Thus, a field-level prediction is insufficient, and providing fine-grained stress maps is more actionable for farmers managing large fields.

Together, these factors present challenges in developing a robust stress detection system that is accurate, scalable, and actionable for farmers.

1.3 Proposed stress detection system

The proposed stress detection system operates in a fully unsupervised manner, with labeled data used solely for evaluation purposes. It functions independently and requires only minimal configuration of file paths, enabling direct application to data from new sugar-beet seasons or years. The system consists of a data preprocessing pipeline that transforms the raw Sentinel-2 temporal images into model-ready data tensors. As part of the preprocessing, sugar-beet fields are divided into smaller sub-patches to address variability in field sizes and enable localized predictions.

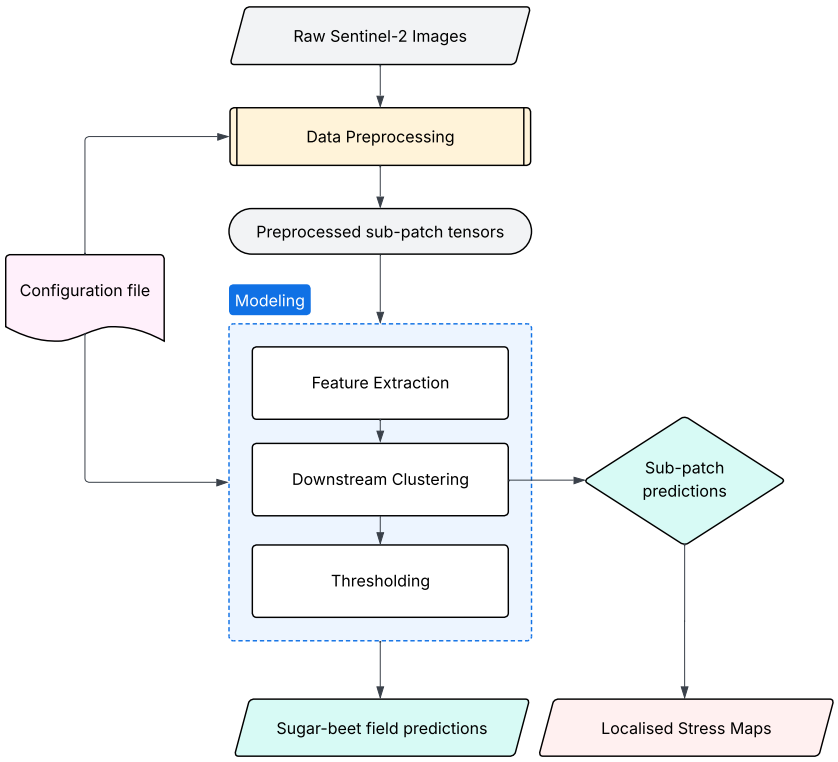


Figure 1.1: Sugar-beet stress detection system workflow.

Subsequently, meaningful representations are extracted from the preprocessed input tensors, followed by downstream clustering to categorize sugar-beet sub-patches as stressed or healthy. A thresholding step aggregates the sub-patch predictions to generate evaluation ready field-level labels. Additionally, sub-patch level predictions are used to produce localized stress maps that highlight stress-affected areas within sugar-beet fields. Figure 1.1 illustrates the end-to-end workflow of the proposed stress detection system, which can be directly applied to new Sentinel-2 data by updating the configuration file, enabling scalable deployment.

The codebase associated with this thesis is available at:
`https://github.com/bhumikasadbhave/Master-Thesis-SITS.git`

2 Background

In this chapter, we discuss the agricultural background of sugar beets including their growth cycle and stress factors. We then discuss different image types and introduce Sentinel-2 imagery. Finally, we explain the concept of vegetation indices and outline the ones used in this thesis.

2.1 Agricultural background

In this section, we provide an overview of sugar-beets, their agricultural significance, growth cycle, and the stress factors affecting their cultivation.

2.1.1 Sugar-beets and their significance in Germany

Sugar beet (Beta vulgaris) is an important crop for the German and the European agricultural sector. The European Union is the world's leading producer of sugar-beet, with around 50% of the total amount [9]. Additional to Germany– Poland, France, and Moldova are also major producers of sugar beet, and thus play a significant role in the European sugar industry (figure 2.1 taken from [50]). These countries are key hubs for sugar-beet cultivation and processing, and they ensure a steady supply for domestic and international demands. Within Germany, Bavaria is one of the key regions where sugar beet cultivation thrives. This is because of the fertile soil, well-established farming infrastructure, and favorable climate.

The sugar-beet industry primarily supports the production of granulated sugar, which is widely used in food products, beverages, and other industrial applications. The sucrose content of the beet is typically around 18-20% when the beet is harvested in autumn, depending on the weather conditions during growth [50]. In addition to sugar, the by-products are also valuable in the livestock feed industry. The major byproducts resulting from sugar-beet processing include — molasses, beet pulp, and beet top silage. Molasses serve as a substrate in yeast production and ethanol fermentation, while beet pulp is a nutritious and cost-effective livestock feed. The crop also plays a role in the biofuel

Figure 2.1: Major sugar-beet producing countries in the EU

industry, contributing to renewable energy production. The benefits of sugar-beet extend beyond the sugar production cycle, as illustrated in figure 2.2 (taken from [51]).

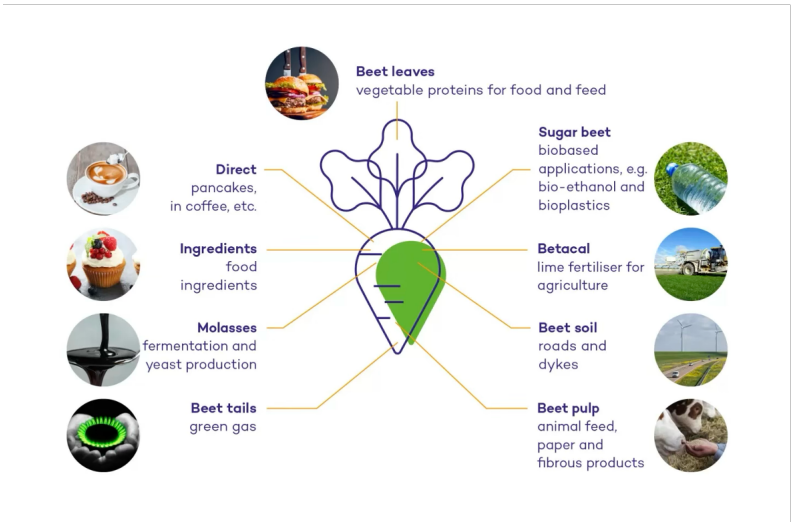


Figure 2.2: Sugar-beet uses

The sugar industry forms a pillar of rural economies in Germany. However, sugar-beet farming has faced increasing challenges, including irregular weather patterns, droughts, and disease outbreaks, which can significantly affect the yields. Thus, the use of advanced monitoring methods using satellite images, has become increasingly important.

2.1.2 Sugar-beet growth cycle and stress factors

The sugar-beet crop has a well-defined growth cycle. The seeds are sowed in early spring (February to March). The plants grow actively during summer, and the leaves grow fully around June. The harvest is done in autumn (mid-September to October). The growth

cycle includes distinct stages: seedling emergence, leaf development, root enlargement, and maturation. Each stage is crucial for achieving high yield and sugar content. The growth cycle is illustrated in figure 2.3 (taken from [51]).

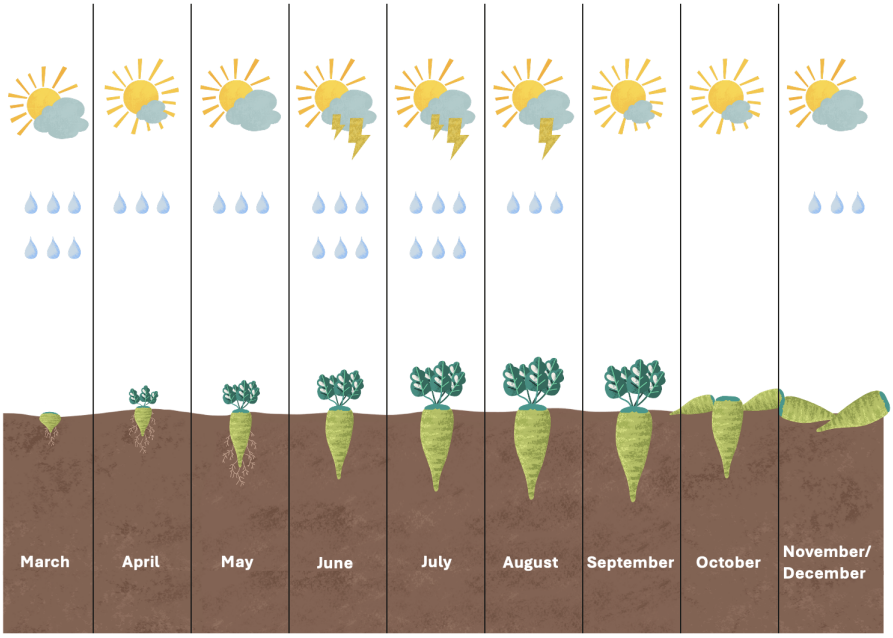


Figure 2.3: Sugar-beet growth cycle

Throughout this lifecycle, sugar-beet crops are subject to a variety of stress factors that can significantly impact both yield and sugar-content. Biotic stress factors, such as fungal and phytoplasma diseases are a persistent threat. Most common diseases include Cercospora Leaf Spot (CLS), Sugar Beet Rust (SBR), and Stolbur phytoplasma. CLS damages leaf tissue, reducing the plant's ability to photosynthesize and grow properly. SBR causes reddish-brown lesions that facilitate the spread of the virus through infected roots and soil. Stolbur phytoplasma affects the plant's nutrient flow, leading to deformed roots and soft, rubbery beets. Figure 2.4 (taken from [18, 15, 7]) shows sugar-beet plants affected by these diseases. Sugar-beet diseases annually reduce profitability by an estimated 10-15%, with losses in individual fields exceeding 50% or more [24]. Disease management strategies, including the use of resistant varieties, crop rotation, and fungicides, are critical but have limitations. Furthermore, since sugar-beet farms are huge, manual inspection of every farm for disease is impractical.

In addition to disease, sugar-beets are also affected by abiotic stress factors. Periods of drought, heat stress, and nutrient deficiencies interfere with the root development and reduce sugar accumulation. These environmental stresses often occur during sensitive growth stages, which increases their negative impact. For example, water scarcity during root enlargement can directly limit biomass accumulation, while excessive heat

Figure 2.4: Sugar-beet diseases - CLS, SBR and Rubbery Beets from left to right

during late maturation can reduce sucrose concentration. As the climate becomes less predictable, abiotic stressors are occurring more often and becoming more severe. This highlights the need for efficient stress detection and monitoring.

2.2 Image modalities

In this section, we give an overview of the types of images used in agricultural applications, including RGB images, multispectral images, and Sentinel-2 satellite data.

2.2.1 RGB images

Ever wondered how the images you click on your phone are created? What are these images composed of? These everyday images are called as RGB (Red-Green-Blue) images, named after the three primary colour channels they have: red, green, and blue. Each channel captures reflections of light at specific wavelengths within the visible spectrum (approximately 400–700 nm). Together, these channels create the vibrant and diverse colours in the image. Each pixel in an RGB image represents a combination of intensities from these three channels. By changing these intensities, millions of colour possibilities can be produced. For example, the violet colour is created by tuning the values of red and blue channels. Figure 2.5 shows an example of RGB image and it's composition (taken from [40]).

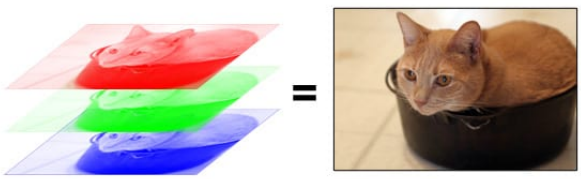


Figure 2.5: RGB image composition

RGB images are at the centre of human visual perception, and they are used extensively in photography, display technologies, and online applications. However, RGB images are limited to the visible spectrum and lack the ability to detect subtle variations in Earth's surface properties, such as plant health or soil moisture. Hence advanced imaging techniques such as multi-spectral imaging are used particularly for agricultural applications.

2.2.2 Multispectral images

Multispectral images go beyond the capabilities of traditional RGB images. These images are acquired using satellites or specialized camera systems, which capture the data across multiple wavelength bands. This includes the wavelengths beyond the visible spectrum, such as the near-infrared (NIR) and shortwave infrared (SWIR). In RGB images, the colours are a result of combining intensities from the red, green, and blue channels. In contrast, the multispectral images have distinct, non-overlapping bands, where each band captures certain properties of the land.

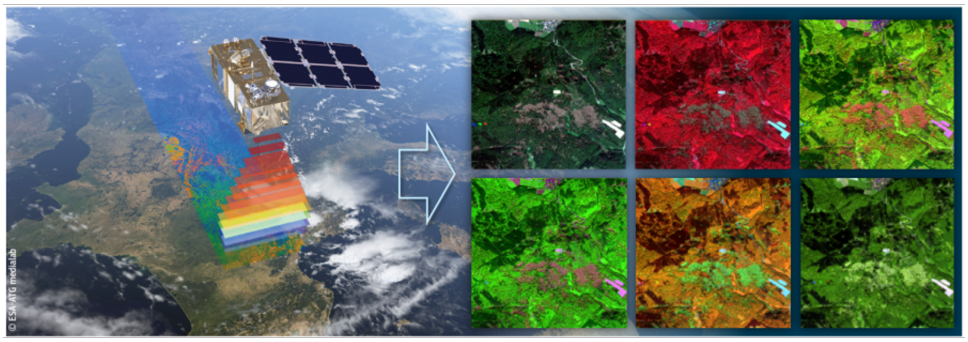


Figure 2.6: Multispectral image

Each band in a multispectral image corresponds to a specific wavelength range, and provides detailed information about the physical and chemical properties of surfaces. For example, visible bands like red, green, and blue capture basic visual features, while NIR bands are sensitive to vegetation health, and SWIR bands detect the moisture content and structural changes. This additional information helps to understand the subtle changes that are invisible to human eyes. Moreover, early signs of plant stress or disease are often manifested due to changes in reflectance in the NIR or SWIR bands before becoming visible in the RGB spectrum [54]. Multispectral images can also be used for temporal analysis, by using the data collected over time, to track changes in crop growth, and monitor disease progression. Figure 2.6 shows an example of a satellite-acquired multispectral image (taken from [44]).

2.2.3 Sentinel-2 satellite images

This thesis uses multispectral images from the Sentinel-2 satellite [14]. The Sentinel-2 is part of the European Space Agency's Copernicus program and is designed to monitor the Earth surface. It provides data at different processing levels, each with distinct properties [45]. The data provided by Greenspin for this thesis is Sentinel Level-2A data. The data at this level is atmospherically corrected, geo-referenced, surface reflectance data. The Sentinel-2 images consist of 13 spectral bands, each corresponding to a specific wavelength range. The bands in the Sentinel-2 Level-2A product include visible, near-infrared (NIR), and shortwave infrared (SWIR) regions, which helps in detailed analysis of various surface properties. The 13 spectral bands of Sentinel-2 images, and their properties (as provided in [43]) are listed in table 2.1, and the overview of Sentinel-2 reflectance ranges is illustrated in table 2.2.

Band Number	Band Name	Wavelength (nm)	Resolution (m/px)
1	Coastal Aerosol	443	60
2	Blue	490	10
3	Green	560	10
4	Red	665	10
5	Red Edge 1	705	20
6	Red Edge 2	740	20
7	Red Edge 3	783	20
8	NIR	842	10
8A	Narrow NIR	865	20
9	Water Vapor	945	60
10	SWIR - Cirrus	1375	60
11	SWIR 1	1610	20
12	SWIR 2	2190	20

Table 2.1: Sentinel-2 spectral bands and their properties

Spectral Range	Wavelength (nm)	Key Characteristics
Visible	400 - 700	Sensitive to chlorophyll absorption
Near Infrared (NIR)	700 - 1100	Sensitive to leaf structure and canopy density
Shortwave Infrared (SWIR)	1100 - 2500	Sensitive to moisture content in vegetation and soil

Table 2.2: Sentinel-2 spectral ranges and their key characteristics

2.3 Vegetaion Indices

Vegetation Indices (VIs) are quantitative measures used to assess the condition of plants or vegetation. They are based on the reflectance values of different spectral bands in the multispectral images. Reflectance refers to the amount of light reflected by a surface compared to the amount of light it receives.

VIs are essentially arithmetic combinations of two or more spectral bands that enhance the reflectance signal of vegetation. Depending on the bands used, a VI highlights a specific property of the vegetation. For example, combining the SWIR and NIR bands gives the Moisture Stress Index (MSI). This VI is used to assess the moisture levels in crops. Vegetation indices play a crucial role in detecting signs of stress or disease in crops. They can detect subtle changes in plants, even before they are visible to the human eye. Diseases such as fungal infections, bacterial pathogens, or viral diseases—often affect the chlorophyll content, water status, and overall health. This can be detected by changes in the reflectance values using VIs.

The VIs used for experimentation in this study include NDVI (Normalised Difference Vegetation Index), EVI (Enhanced Vegetation Index), and MSI (Moisture Stress Index). The following sub-sections dicuss these Vegetation Indices in detail, and why they are suitable for stress detection in sugarbeets.

2.3.1 Normalised Difference Vegetation Index (NDVI)

The Normalised Difference Vegetation Index (NDVI) [61] is one of the most commonly used indices for assessing vegetation health. It works on the principle that the healthy vegetation absorbs most of the visible light (specifically red light) for photosynthesis, and strongly reflects near-infrared (NIR) light. NDVI is calculated as follows:

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}} \ , \tag{2.1}$$

where Red = Sentinel Band 4, and NIR = Sentinel Band 8 (Near Infrared).

NDVI values range from –1 to +1, where higher values indicate dense healthy vegetation, values near zero correspond to sparse or stressed vegetation and bare soil, and negative values represent non-vegetated surfaces like water or clouds.

Relevance to sugar-beet stress: NDVI is used for detecting general vegetation stress. In the context of sugar-beet fields, diseases like SBR (Sugar-beet Rust) and Cercospora,

along with environmental factors such as drought, nutrient deficiency, and extreme temperatures, can cause significant chlorophyll degradation and reduce the photosynthetic efficiency of plants. These changes are effectively captured by NDVI. However, NDVI values can saturate in areas with very high vegetation density, and in such cases, further differences in chlorophyll content are not detectable. Thus, NDVI works best only as an overall indicator of vegetation health. It is commonly used to detect vegetation and forest degradation [30].

2.3.2 Enhanced Vegetation Index (EVI)

The Enhanced Vegetation Index (EVI) [58] improves upon NDVI by reducing the influence of soil background and atmospheric conditions. It uses reflectance from the Sentinel-2 NIR, red, and blue bands. EVI is particularly effective in areas with high vegetation density, where NDVI may saturate. It is calculated as follows:

$$\text{EVI} = G \cdot \frac{\text{NIR} - \text{Red}}{\text{NIR} + C_1 \cdot \text{Red} - C_2 \cdot \text{Blue} + L} \ , \tag{2.2}$$

where, the values for Sentinel-2 data as provided in [58]:
G = gain factor (2.5),
C_1 and C_2 = coefficients for the Red and Blue bands (6 and 7.5, respectively),
L = canopy background adjustment value (1),
NIR = Sentinel Band 8, Red = Sentinel Band 4, and Blue = Sentinel Band 2

The EVI values range from -1 to +1, where higher values indicate healthy vegetation, and lower values correspond to sparse or stressed vegetation or bare soil. Unlike NDVI, EVI is designed to be more resistant to atmospheric interference and background effects due to soil [17].

Relevance to sugar-beet stress: The Enhanced Vegetation Index (EVI) helps monitor the health of sugar-beet fields by detecting signs of stress in the plants. Conditions like Sugarbeet Rust (SBR) and Cercospora Leaf Spot (CLS), as well as environmental stressors such as drought and heat stress, often cause changes in the leaf structure and pigment, which lead to lower EVI values.

2.3.3 Moisture Stress Index (MSI)

The Moisture Stress Index (MSI) [32] measures plant water content by comparing reflectance in the SWIR1 and NIR bands. It is highly sensitive to changes in plant water

status. Thus it is most commonly used for detecting moisture stress in crops. It is calculated as follows:

$$\text{MSI} = \frac{\text{SWIR 1}}{\text{NIR}} \ , \tag{2.3}$$

where SWIR 1 = Sentinel Band 11 and NIR = Sentinel Band 8

MSI typically ranges from 0 to more than 3, with higher values indicating vegetation experiencing greater stress due to reduced moisture content in the leaves. The common range for green vegetation is 0.4 to 2 [33].

Relevance to sugar-beet stress: MSI is useful for detecting Rubbery Beets, a disease caused by Stolbur Phytoplasma. This disease damages the plant's system for moving water and nutrients, leading to wilting, leaf dehydration, and water retention in the roots- making them soft and rubbery. In addition, environmental factors like prolonged drought, soil salinity, and heat stress can also affect the water content, contributing to similar stress signatures detectable by MSI.

A single vegetation index captures only a specific aspect of the vegetation, and may not be sufficient by itself. Hence, multiple indices are often used together for more accurate stress and disease detection [29],[20]. Therefore, in this thesis, we use the three vegetation indices — NDVI, MSI, and EVI in combination for experimentation (section 6.3.2).

3 Related work

This chapter provides a brief overview of relevant literature, followed by the key concepts that support the methods used in this thesis.

3.1 Literature review

Traditional manual inspection of crops is time-consuming, labor-intensive, and prone to human error, specially across large agricultural areas. In contrast, images captured by satellites or drones provide a scalable and efficient way to observe crop conditions and detect stress. With the growing availability of such data, many studies have explored the use of spectral information for identifying plant stress and disease.

3.1.1 Machine learning techniques

A study [46] uses statistical data features and basic image processing for disease detection in rice and wheat crops. It uses data from the NASA terra [34] and ISRO RISAT-I [23] satellites. The authors use images of healthy and diseased leaves to set threshold values. Satellite images that exceed these thresholds are then analyzed using histogram analysis and edge detection to identify signs of disease.

A study [37] study explores the potential of using high-resolution (3 meters) PlanetScope satellite images [35] for detection of sudden death syndrome in soybean plants. It uses multiple bands—including blue, green, red, and near-infrared (NIR)—along with the NDVI vegetation index to assess plant health. The ground truth data for this study was collected manually from soybean fields. The study uses the Random Forest classification algorithm to categorise the fields as either healthy or diseased. The findings suggest that spectral data can help identify risk areas in soybean fields effectively.

Another study [41] uses Support Vector Machines (SVMs) for detection and categorisation of sugar-beet diseases. It uses 9 different spectral Vegetation Indices, and detects and classifies 3 different sugar beet diseases — Cercospora, Leaf Rust and Powdery

Mildew. The experiments were performed on sugar beet plants in a Greenhouse, and multispectral reflectance data was captured using a handheld spectro-radiometer. In addition, RGB images were taken using handheld cameras for analysis. The study showcases the advantage of using combinations of various Vegetation Indices for disease detection in sugar-beet.

Inspired by the study [46], we extract histogram features from the data for downstream clustering. Similarly, drawing from studies [37] and [41], we explore the use of Vegetation Indices suited for our use-case, and compare their performance with that of raw spectral bands.

The studies discussed in this subsection highlight the use of satellite images for disease detection. Some use basic image processing techniques, while others use machine learning algorithms to achieve their objectives. However, traditional machine learning methods may struggle due to the high dimensionality of satellite data, making it important to explore deep learning-based approaches.

3.1.2 Deep learning strategies

A review [56] analyses 193 studies, and explores the use of deep learning on satellite images across agricultural tasks. It finds that Convolutional Neural Networks (CNNs) outperform traditional machine learning methods in crop classification. However, while deep learning is widely applied in land use and land cover mapping, its use in other areas like disease detection is still limited. The review attributes this mainly to the scarcity of labeled datasets.

A study [68] uses satellite images for detection of Pine Wilt Disease (PWD), a widespread forest disease that causes the death of pine species. The study uses an unmanned aerial vehicle (UAV) equipped with multispectral cameras to obtain images. Through site visits and surveys, the images were labeled manually using the Label Studio software [27]. Two disease detection algorithms— R-CNNs (Region-based Convolutional Neural Networks [19]) and YOLOv4 (You Only Look Once version 4 [8]) and two traditional machine learning algorithms—Random Forest and Support Vector Machines were employed. Results showed that the deep learning models outperformed the traditional ones, enabling early detection of infected pine trees.

Another study [25] introduces Temporal CNNs (TempCNNs), which uses 3D convolutions [1] for crop classification using temporal satellite images. The study presents a 3D convolutional kernel, a 3D CNN architecture, and an active learning strategy for improved crop classification. It adapts the widely used VGGnet architecture (Visual Geometry Group [47]), and replaces the traditional 2D convolution operations with 3D convolutions. It uses the Gaofen-1 and Gaofen-2 images [59] with 4 spectral bands, and 3

temporal instances per image. The results demonstrate that the 3D CNNs outperforms 2D CNNs, showing that 3D convolutions are more effective at extracting features, as they preserve the temporal information that 2D convolutions might overlook.

Inspired by the study [25], we explore the concept of 3D convolutions for temporal Sentinel-2 satellite images, however in an unsupervised way using representation learning.

Although satellite images are abundant and well suited for data-hungry deep learning models, obtaining accurate ground truth labels remains a significant challenge. The studies [68, 25] approach disease detection as a classification task mainly using CNNs, with labels collected manually. While these supervised methods perform well, they can be impractical for small-scale farmers or industries lacking resources to label large datasets. This highlights the need to explore approaches that work with limited or no labeled data.

3.1.3 Respresentation learning techniques

A study [28] presents an unsupervised feature extraction framework for hyperspectral images using MLP-based autoencoders. The approach explores both single-layer and multi-layer stacked autoencoders, as well as the impact of applying Principal Component Analysis (PCA) to spectral channels before feature extraction. The extracted features are then used for downstream classification. The experiments were performed on a UAV-acquired hyperspectral data with 13 land cover categories, and 224 spectral bands. The results indicate that autoencoder based features outperform traditional methods like SVM and PCA-based SVM, demonstrating their effectiveness for high-dimensional data.

The study [70] takes advantage of the temporal nature of hyperspectral images for change detection. It proposes a spectral-temporal autoencoder, which extracts features from both spectral and temporal dimensions by using 3-dimensional convolutions. The autoencoder is trained on the unlabeled 244 band images— taken with the airborne visible infrared imaging spectrometer sensor. The labeled samples are used for the downstream classification task. Compared with the state-of-the-art change detection algorithms, experimental results verify the effectiveness of 3D convolutions.

The studies [28] and [70] use high-dimensional hyperspectral data (200+ bands) to train autoencoders. While [28] employs MLP-based autoencoders, [70] leverages the temporal aspect of satellite data using 3D convolutions. However, their applicability to multispectral data like Sentinel-2 images with only 13 bands remains unclear.

Inspired by [70] and [25], we explore representation learning on temporal Sentinel-2

images using autoencoders with both 2D and 3D convolutions. In contrast to these studies, which perform downstream classification, we adopt a downstream clustering approach due to limited labeled data.

3.1.4 Self-supervised learning methods

Recent advances in self-supervised learning have greatly impacted remote sensing. The review [57] summarizes key self-supervised learning methods, categorizes them, and highlights important works. It also analyzes the application of these techniques on widely used remote sensing datasets.

A study [53] introduces the Temporo-Spatial Vision Transformer (TSViT), a fully-attentional model designed for SITS data. This model is based on Vision Transformers (ViTs) [16] and draws inspiration from previous work that applies ViT for video processing [6]. To enhance the model performance, acquisition-time-specific temporal encodings and multiple learnable class tokens are incorporated. This approach outperforms the benchmarked methods on SITS datasets for land cover classification.

Masked Autoencoders [21] extend Vision Transformers by employing separate Transformer based encoder and decoder components. The SatMAE framework [10] adapts this approach specifically for temporal and multispectral satellite data, using unlabeled Sentinel-2 images. SatMAE features an asymmetric encoder-decoder architecture and incorporates both positional encodings and acquisition-date-specific temporal encodings. This method demonstrates significant improvements over previous state-of-the-art techniques in downstream supervised and transfer learning tasks.

Studies [53], [21] and [10] show how Vision Transformer-based models can learn data representations effectively. They also highlight the importance of using temporal encodings, which is particularly important when working with SITS datasets. Inspired by these studies, we incorporate acquisition-date-specific temporal encodings for the proposed Autoencoder-based models.

Additionally, inspired by [10] we attempt to develop a framework similar to temporal-SatMAE, adapted for the dataset and problem setting of this thesis.

3.2 Conceptual foundations

In this section, we provide the theoretical background for the baseline and main models used in this thesis. It includes the techniques for feature engineering, dimensionality

reduction, representation learning, and convolutional operations.

3.2.1 Feature engineering and dimensionality reduction

The traditional approach relies on handcrafted features derived from pixel-level statistics to summarize the most informative aspects of the data. Common techniques include summary statistics such as mean, standard deviation, and histogram-based features. In this subsection, we discuss two techniques: histogram-based features for capturing intensity distributions, and PCA for dimensionality reduction.

Histogram features

Histogram features [64] are statistical representations of the distribution of numerical data (pixel intensities in case of images) by grouping values into a set of intervals called *bins*. To construct a histogram, the data range is partitioned into k non-overlapping, adjacent intervals $[b_0, b_1], [b_1, b_2], \ldots, [b_{k-1}, b_k]$, which are typically equal in width. For each bin i, a count m_i is computed, representing the number of data points falling within that interval. Given a dataset $I = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}$, the histogram values satisfy:

$$n = \sum_{i=1}^{k} m_i \ , \tag{3.1}$$

where k is the total number of bins and n is the total number of observations. Formally, the histogram feature vector $\mathbf{h} = (h_1, \ldots, h_k) \in \mathbb{R}^k$ is defined as:

$$h_i = |\{x \in I \,|\, b_{i-1} \leq x < b_i\}| \quad \text{for } i = 1, \ldots, k \tag{3.2}$$

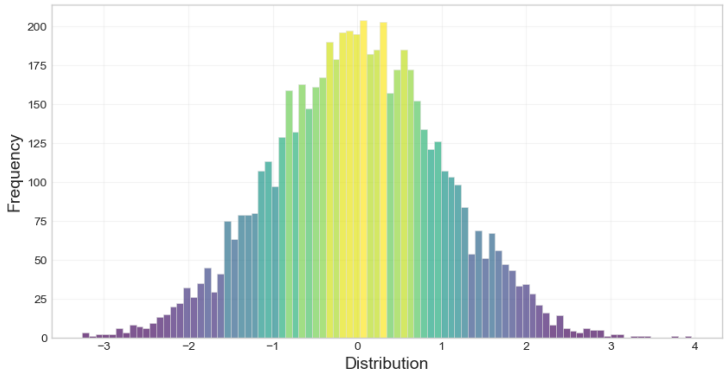


Figure 3.1: Histogram example

Histogram features are simple to compute, compact, and effective for capturing global spectral characteristics. However, the limitation of histogram features is that it ignores spatial relationships within the image. Figure 3.1 shows an example histogram of values randomly drawn from a standard normal distribution.

Principal Component Analysis (PCA)

PCA [65] is a widely used dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving as much variance as possible. By identifying orthogonal directions (principal components) along which the data varies the most, PCA helps to capture the most important patterns in the data.

Given a data matrix $X \in \mathbb{R}^{n \times d}$ (with n samples and d features), PCA computes the eigenvectors of the covariance matrix $C = \frac{1}{n-1} X^\top X$, assuming X has been mean-centered. We then compute the eigenvectors of C, and select the top k eigenvectors to form a projection matrix $W \in \mathbb{R}^{d \times k}$. The transformed data is obtained by:

$$Z = XW \ , \tag{3.3}$$

where $Z \in \mathbb{R}^{n \times k}$ is the reduced representation that captures the most significant variance in the data.

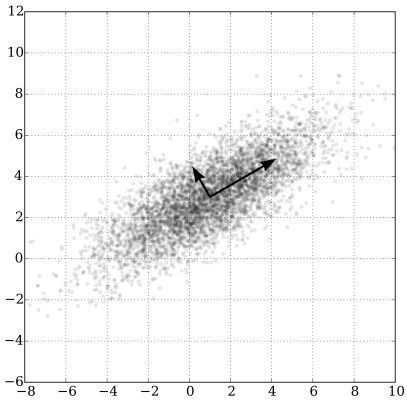


Figure 3.2: PCA example

Figure 3.1 (taken from [65]) illustrates the PCA of a multivariate Gaussian distribution. The vectors represent the eigenvectors of the covariance matrix, each scaled by the square root of its corresponding eigenvalue.

3.2.2 Representation learning and autoencoders

Representation learning [38] is a concept that focuses on discovering useful features or patterns from raw data. Instead of relying on hand-crafted features, representation learning methods learn transformations of the input data into formats (or representations) that make downstream tasks like classification or clustering more effective and efficient. These representations capture the underlying structure, semantics, or variations in the data that may not be obvious in its raw form.

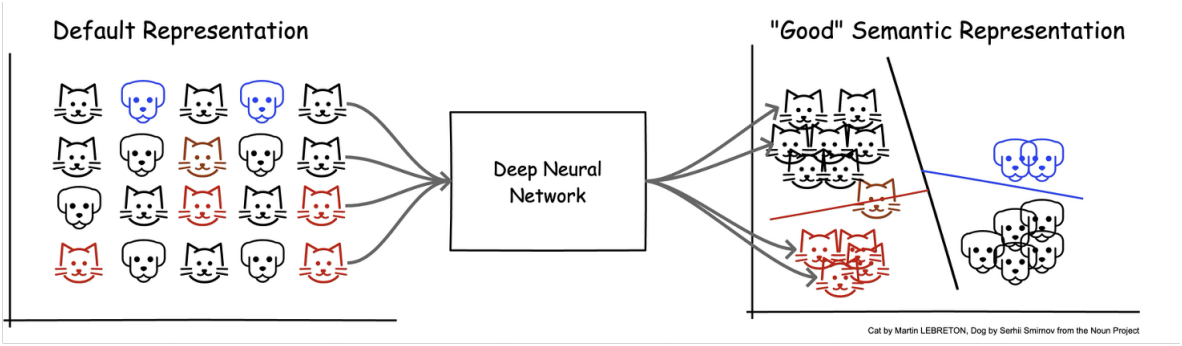


Figure 3.3: An example of how representation learning can help in creating a better representation of the data.

As illustrated in figure 3.3 (taken from [39]), representation learning can transform complex data distributions into more separable and structured spaces, making it easier for machine learning models to interpret the data.

Autoencoders

Autoencoders are neural networks designed for unsupervised representation learning tasks, such as dimensionality reduction, feature extraction, and data compression. Unlike supervised learning models that rely on labeled data, auto-encoders learn to encode data into a lower-dimensional space and then decode it back to its original form without requiring any labels. The goal of an autoencoder is to learn an efficient representation of the input data by capturing the most important features of that data while discarding unnecessary details.

An autoencoder consists of two primary components:

Encoder: The encoder is the first part of the network. It is responsible for taking high-dimensional input data $\mathbf{x}$ and compressing it into a lower-dimensional representation $\mathbf{z}$, also known as the latent space or bottleneck. During this process, the encoder learns the most important features of the input, and also the underlying patterns in the data.

Decoder: The decoder takes the latent bottleneck $\mathbf{z}$ produced by the encoder and reconstructs it back to its original form, $\hat{\mathbf{x}}$. The $\hat{\mathbf{x}}$ should closely match the input data $\mathbf{x}$. The goal of the decoder is to use the compressed information to recreate the original data as accurately as possible. The decoder can also be thought of as a mirror image of the encoder, with layers that expand the data back to its original dimensions.

The overall process can be represented as:

$$\hat{\mathbf{x}} = \text{Decoder}(\text{Encoder}(\mathbf{x}))$$

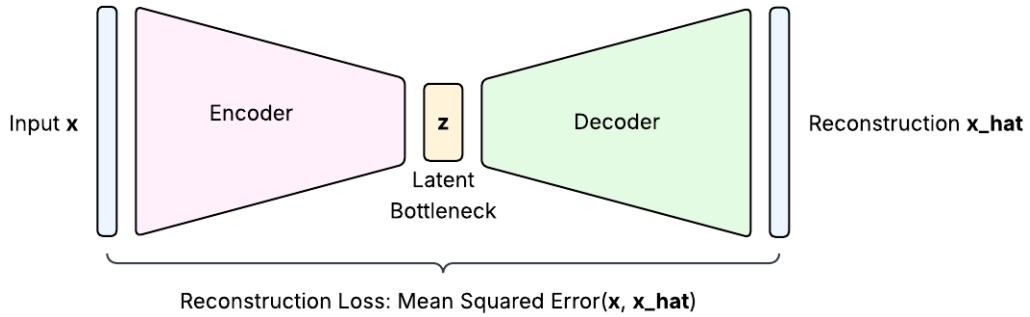


Figure 3.4: Autoencoder Architecture

Loss function: The goal of autoencoder is to minimise the difference between the input data and the reconstructed output. This difference is typically measured using a loss function, such as Mean Squared Error (MSE) or Binary Cross Entropy (BCE), depending on the task and nature of the data. This loss is called as the *Reconstruction Loss*. The most commonly used loss function is the MSE loss, which is defined as:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \quad , \tag{3.4}$$

where $\mathbf{x}$ is the input data, $\hat{\mathbf{x}}$ is the reconstructed data, and n is the number of examples.

The network updates its parameters (weights) through backpropagation to minimise this reconstruction loss. During this process, the network learns a compact and meaningful representation of the data in the latent bottleneck. The architecture of a basic autoencoder is illustrated in figure 3.4, with $\hat{\mathbf{x}}$ represented as **x_hat**.

3.2.3 Convolutions refresher

Convolutional operations are fundamental building blocks in modern deep learning architectures. They are important specifically for spatiotemporal data analysis. They

are used to extract local spatial patterns from structured data like images and video frames.

1D convolutions

1D convolutions are typically used for sequential data, such as time-series, sensor readings, or text sequences. 1D convolutions operate along a single dimension (e.g., time or position), and extracts the local patterns.

Let the input sequence be $\mathbf{x} \in \mathbb{R}^L$ and the kernel $\mathbf{k} \in \mathbb{R}^l$, where l is the kernel size. The 1D convolution output $\mathbf{y} \in \mathbb{R}^{L_{\text{out}}}$ is computed as:

$$\mathbf{y}_i = \sum_{j=0}^{l-1} \mathbf{k}_j \cdot \mathbf{x}_{i+j} \tag{3.5}$$

Figure 3.5 (taken from [5]) shows a simple example of 1D Convolution.

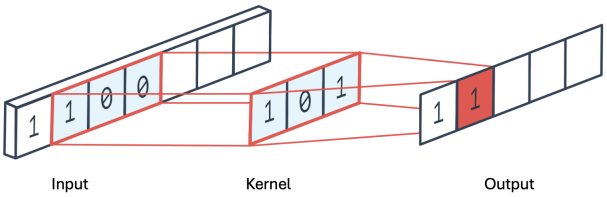


Figure 3.5: 1D convolution example

1D convolutions efficiently model local dependencies in sequences and have lower computational costs than 2D or 3D convolutions. They are well-suited for capturing temporal patterns, signal processing, or text embeddings. However, since they operate along only one dimension, they cannot capture spatial or multi-dimensional patterns.

2D convolutions

2D convolutions are a fundamental building block for modeling images. They operate by sliding a small filter or kernel across the height and width of an image to detect local spatial patterns such as edges, corners, textures, and shapes. This localized operation allows the model to preserve the spatial arrangement and structure of features in the image, which is essential for tasks like object detection and image classification.

For an input matrix $X \in \mathbb{R}^{H \times W}$ and a kernel $K \in \mathbb{R}^{k \times k}$, where k is the spatial size of the kernel, the 2D convolution output $Y \in \mathbb{R}^{H' \times W'}$ is computed as:

$$Y_{ij} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} K_{mn} \cdot X_{i+m,\, j+n} \tag{3.6}$$

Figure 3.6 (taken from [31]) shows a simple example of how 2D Convolutions work.

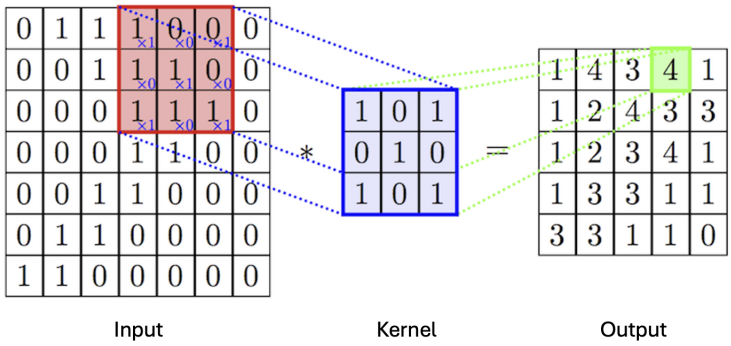


Figure 3.6: 2D convolution example

2D convolution is computationally efficient and effective at extracting spatial features. It is widely used in many frameworks and pretrained models. However, since the 2D kernel only operates over height and width, 2D convolution does not capture temporal dependencies, making it less suitable for time series data.

3D convolutions

When working with temporal or spatiotemporal data such as videos or satellite image sequences, it is important to model dependencies across time. 2D convolutions treat each frame independently and thus, fails to capture motion or changes across time. 3D Convolutions extend 2D convolutions by adding an extra temporal dimension to the input and to the convolution kernel. A 3D kernel slides not only spatially (height and width) but also across time (depth), and does feature extraction from consecutive frames.

Let the input tensor $X \in \mathbb{R}^{T \times H \times W}$ and the kernel $K \in \mathbb{R}^{t \times k \times k}$, where t is the temporal size and k is the spatial size of the kernel. The 3D convolution output $Y \in \mathbb{R}^{T' \times H' \times W'}$ is computed as:

$$Y_{tij} = \sum_{p=0}^{t-1} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} K_{pmn} \cdot X_{t+p,\, i+m\, j+n} \tag{3.7}$$

Figure 3.7 (taken from [67]) illustrates a simple example of 3D Convolution.

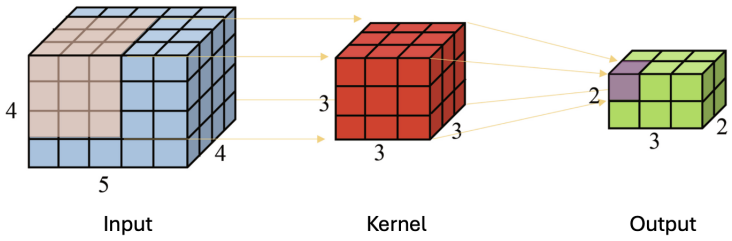


Figure 3.7: 3D convolution example

3D convolutions jointly capture spatial and temporal features, making them ideal for tasks involving motion, dynamics, or sequential data—such as videos or satellite image time series. However, they come with higher computational and memory costs compared to 2D convolutions.

Note: For simplicity, the convolution operations above are presented without explicitly including the channel dimension. In practice, convolutional layers operate over multiple input and output channels, which can be incorporated by extending the formulae accordingly. A separate kernel is used for each input-output channel pair, and the results are summed to produce the final output feature map. The simplified notation in this section is intended to aid visual understanding.

4 Data

In this chapter, we describe the dataset and the preprocessing pipeline that transforms raw Sentinel-2 satellite images into model-ready tensors.

4.1 Dataset

This study focuses on data spanning from June to early September. This period covers the sugar-beet growth cycle, beginning when the foliage has grown enough to cover the ground and extending until harvest. The sugar-beet fields under analysis are divided into two regions, R1 and R2, each supplying independent production plants (figure 4.1). Located in geographically distinct parts of Germany, these regions differ in soil types and climatic conditions. In this thesis, we primarily use fields from the 2019 dataset that belong to region R1. Given that agricultural fields are influenced by climatic variability such as rainfall and soil differences, we further apply our stress detection system on data from region R2 for the year 2024 in the experiments section.

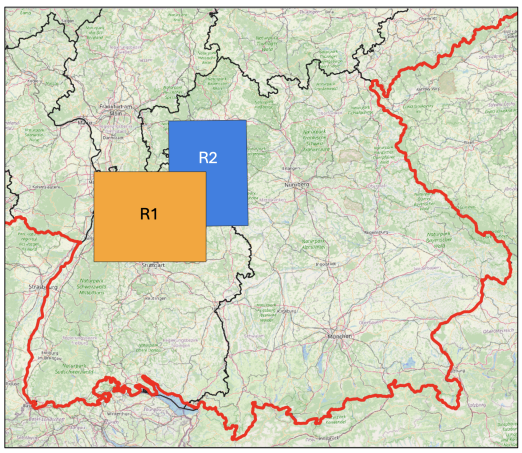


Figure 4.1: Geographic regions R1 and R2 in Germany, corresponding to the 2019 and 2024 datasets respectively.

The evaluation set for the 2019 season includes 35 stressed and 26 healthy sugar beet fields. In 2024, the set contains 35 stressed fields but only 5 healthy ones—largely due

to heavy rainfall and increased disease outbreaks. Moreover, class imbalance remains a potential concern—both in the unlabeled training sets for 2019 and 2024, and in future datasets. This imbalance may arise from two primary factors. First, environmental conditions can lead to widespread stress across fields, making healthy cases relatively rare. Second, ground-truth labels are primarily collected by farmers, who tend to report stressed fields more frequently due to their agronomic relevance. As a result, the datasets may potentially over-represent stressed cases.

4.1.1 Spatial attributes

For this study, we use Sentinel-2 Level 2A [13] satellite images provided by Greenspin, focusing on agricultural regions with sugar beet cultivation. There are multiple sugar beet fields present in each image. The images have spatial dimensions of $(1000, 1000)$ pixels, where each pixel represents a float32 reflectance value. The reflectance values range from 0 to 1, and can reach values up to 2 in some scenarios— for example, dry sand exhibits high reflectance in Sentinel band 4 (Red) [69].

4.1.2 Spectral attributes

The Sentinel-2 satellite provides 13 spectral bands spanning the visible (Blue, Green, Red), near-infrared (NIR), and shortwave infrared (SWIR) regions of the electromagnetic spectrum. These bands vary in spatial resolution ranging from 10 to 60 meters, and each captures distinct surface characteristics.

For this study, the Sentinel-2 Bands 1, 9 and 10 are excluded from the images provided by Greenspin. Band 1 (Aerosol-60m) is mainly used for coastal and aquatic applications, and it helps in detecting suspended particles and correcting atmospheric effects over water bodies. For analysing sugar beet fields, Band 1 offers limited value since it is not sensitive to soil and plant characteristics. Moreover, since we have a reliable cloud mask provided by Greenspin, the water vapour correction of Band 9 and cirrus detection of Band 10 become redundant. Hence, Band 9 (60m) and band 10 (60m) are also excluded. The selected bands are stored in a numpy array with each band occupying a separate channel, indexed from 0, as detailed in table 4.1.

Resmapling channels to 10m resolution

The visible and near-infrared bands (Bands 2, 3, 4, and 8) of Sentinel-2 have a native resolution of 10 meters, while the remaining bands used in this study (Bands 5, 7, 8A,

Channel Index	Sentinel-2 Band	Original Resolution (m/px)	Band Information
0	Band 2	10	Blue
1	Band 3	10	Green
2	Band 4	10	Red
3	Band 5	20	Red Edge 1
4	Band 6	20	Red Edge 2
5	Band 7	20	Red Edge 3
6	Band 8	10	Near Infrared (NIR)
7	Band 8A	20	Narrow Near Infrared (NIR)
8	Band 11	20	SWIR 1 (Shortwave Infrared)
9	Band 12	20	SWIR 2 (Shortwave Infrared)
10	Cloud Mask		Mask for Clouded Pixels
11	Sugar-beet Field ID Mask		Mask to identify Sugar-beet field pixels and their IDs
12	Acquisition Date Mask		Mask containing acquisition date of the image

Table 4.1: Mapping of Sentinel-2 bands and masks to channel indices

11, and 12) have a native resolution of 20 meters. To ensure uniform spatial resolution across all bands and streamline the preprocessing, Greenspin has resampled the 20-meter bands to a 10-meter resolution.

4.1.3 Temporal attributes

For every Sentinel-2 image capturing a unique scene, there are multiple temporal images that capture the same scene. We refer to these as the *temporal instances* of an image throughout this thesis. There are an inconsistent number of temporal instances per image. They span from June 1st to September 9th, aligning with the growth cycle of sugar-beet plants. During this period, the leaf canopy is typically dense enough to minimize the influence of background soil in the Sentinel-2 images.

This thesis leverages this temporal nature of data to study disease progression and identify fields that exhibit abnormal growth patterns due to stress (including disease induced stress), as compared to the growth pattern of healthy fields. Since every image has different number of temporal instances, we temporarily denote the number of temporal instances as T for the initial phase of preprocessing.

An individual data sample with its associated temporal instances is refered to as *Image*

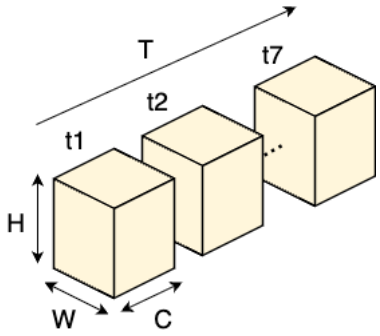


Figure 4.2: Image data represented as a sequence of temporal instances from t1 to t7. Each cube corresponds to a single instance with dimensions (H,W,C).

Mask	**Purpose**	**Consistency across temporal instances**
Cloud	Remove cloud-affected images	In-consistent
Sugar-beet field ID	Track field numbers and remove non-sugar-beet pixels	Consistent
Acquisition date	Track acquisition dates and create temporal encodings	In-consistent

Table 4.2: Overview of masks used and their consistency across the temporal dimension.

Tensor in this thesis, and has the dimensions $\mathbb{R}^{1000\times1000\times10\times T}$, with $(1000, 1000)$ spatial size, 10 channels, and T number of temporal instances. In Step 3 of the preprocessing pipeline, we select seven temporal instances for each image. Figure 4.2 illustrates the structure of an individual data sample with these seven temporal instances.

4.1.4 Masks

This thesis uses several masks in addition to the Sentinel-2 images. They simplify preprocessing and enable tracking of relevant information. At the end of the preprocessing pipeline, these mask channels are removed. An overview of the masks is provided in table 4.2. Additionally, to understand the relationship between the images and the masks, an Entity-Relationship diagram is provided in figure 4.3.

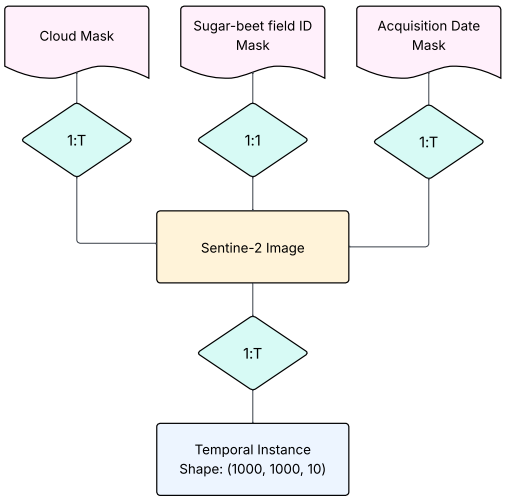


Figure 4.3: Entity-Relationship diagram for Sentinel-2 Image. The relationship 1:1 indicates one-to-one mapping between the entities, while 1:T indicates one-to-many mapping- with T equals the number of temporal instances associated with the corresponding image.

Cloud mask

The cloud mask provided by Greenspin is derived from the Sentinel-2 Scene Classification Map [22]. This mask is used to eliminate pixels with any cloud probability, shadows, or defective pixels. Specifically, it excludes the pixels belonging to the classes below:

- No Data (Missing data)
- Saturated or defective pixel
- Dark features or shadows
- Cloud shadows
- Cloud medium probability
- Cloud high probability
- Thin cirrus: Cirrus clouds are short, detached, clouds found at high altitudes
- Snow or ice

This mask is different across all temporal instances of an image, since the cloud coverage depends on the acquisition time.

Sugar-beet field ID mask

The sugar-beet field ID mask consists of the unique IDs of the sugar-beet fields. This mask is provided by Greenspin for field identification and tracking. It consists of the field

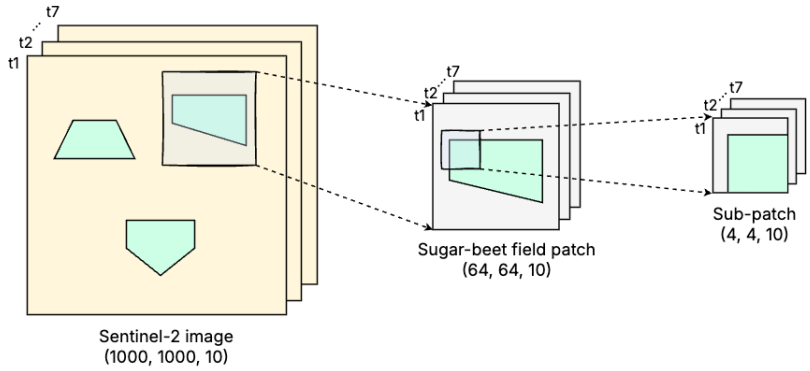


Figure 4.4: Data preprocessing overview

number as a floating-point value (example: 1230818.0). Pixels belonging to a sugar-beet field are assigned this value, while non-field pixels are assigned 0. Unlike the cloud mask, this field mask is constant across all temporal instances of a given image.

Acquisition date mask

The acquisition date mask consists of the acquisition date of the image for pixels belonging to sugar-beet fields, with all other pixels assigned a value of 0. During data preparation, acquisition dates are used to select temporal instances and create uniform-length temporal stacks for the sugar-beet fields. They also facilitate generating temporal encodings, which are essential during the modeling phase. Since each temporal image has a unique acquisition timestamp, the mask differs for each temporal instance. We generate this mask during the preprocessing phase by extracting the acquisition dates from the filenames of the temporal instances.

All three discussed masks are integrated into the images as additional channels-10, 11, and 12 (table 4.1). Including these masks simplifies preprocessing and facilitates tracking of field IDs and acquisition dates. The mask channels (10, 11, and 12) are removed at the end of preprocessing. The modified *Image Tensor* consists of the selected Sentinel-2 bands and the masks, having the dimensions $\mathbb{R}^{1000\times1000\times13\times T}$, with $(1000, 1000)$ spatial size, 13 channels, and T number of temporal instances.

4.2 Data preprocessing

In this section, we describe the processing of raw Sentinel-2 images to generate model-ready data tensors. Figure 4.4 provides an overview of the preprocessing, while figure 4.5

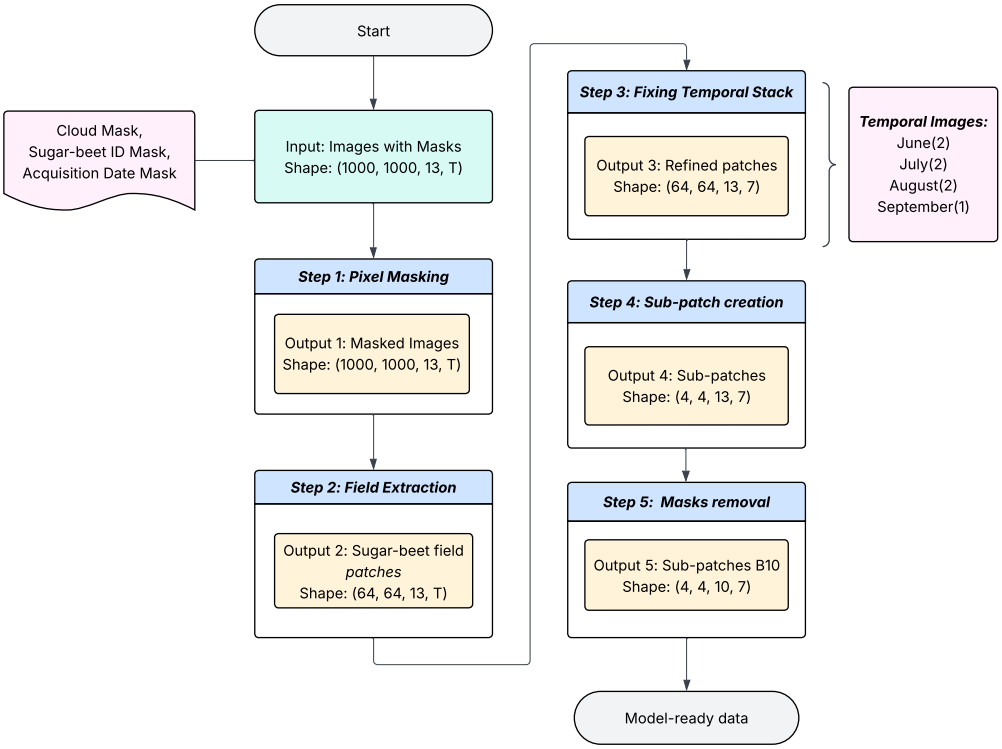


Figure 4.5: The data preprocessing pipeline with corresponding output shapes. For temporal images, the number in braces indicates the number of temporal instances chosen.

details the step-by-step pipeline. The following subsections provide an in-depth explanation of each preprocessing step. The input to the pipeline is the Sentinel-2 *Image Tensor* with integrated masks.

4.2.1 Pixel masking

As the first preprocessing step, we mask out invalid pixels—those not belonging to sugar-beet fields—from the Sentinel-2 images. We binarize the sugar-beet field ID mask and apply it pixel-wise across all channels and temporal instances of the input data. This process removes pixels outside the sugar-beet fields, retaining only the relevant field pixels. The output of this step, referred to as *Output 1*, maintains the same dimensions as the input to the preprocessing pipeline:

Output 1 Shape: $\mathbb{R}^{1000\times1000\times13\times T}$

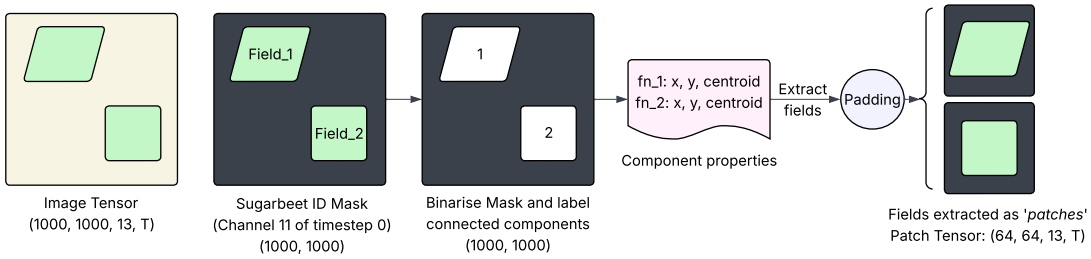


Figure 4.6: Sugar-beet field extraction using the field ID mask. The 4D tensor is visualized as 2D for simplicity.

4.2.2 Extracting the sugar-beet fields

The next preprocessing step is extraction of the individual sugar-beet fields. For better understanding, the process for a single image is illustrated in the figure 4.6.

The sugar-beet field ID mask is used to extract sugar-beet fields from the images. First, the mask is binarized by converting all values greater than 0 to 1, while keeping zeros unchanged. We then identify and label distinct connected components (regions) in the binary image. A unique integer label is assigned to each connected component. A connected component is defined as a group of 8-connected neighboring pixels that share the same value (in this case, value = 1) [2]. Figure 4.7 (taken from [12]) illustrates the different types of pixel connectivity. A labeled image is created, where each connected component is assigned a unique integer label. The function *label* from *skimage.measure* library is used for assignment [49]. Algorithm 1 details the working of this function.

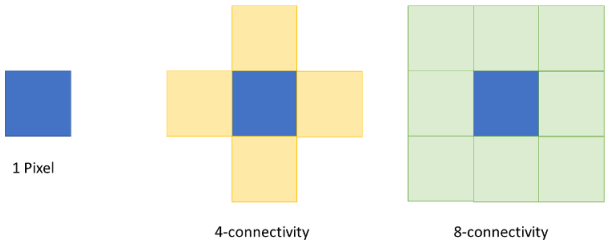


Figure 4.7: Pixel connectivity types in 2D

Once connected components are labeled, the properties of each component such as the coordinates, and the centroid are extracted. This is done using the *regionprops* function from skimage.measure library [48]. Using these properties, a bounding box for every component is created, which is essentially the smallest rectangular box that encloses the entire connected component in the image. Then, for each labeled region, the bounding box coordinates are retrieved, and a patch of the image corresponding to that region is extracted accross the corresponding temporal instances. Since connected components

Data: A binary mask with components
Result: A labeled mask with unique integers for each connected region
Initialize `labeled_mask` with 0s and set `current_label` ← 1;
foreach *pixel* (row, col) *in* `binary_mask` **do**
    if `binary_mask[row,col]==1` **and** `labeled_mask[row,col]==0` **then**
        Perform connected-component labeling:;
            Assign `current_label` to all 8-connected pixels;
        Increment `current_label`;
    end
end
return `labeled_mask`;

Algorithm 1: Connected-component labeling of a binary mask

are used to extract sugar-beet field patches, adjacent fields may be merged into a single component. In such cases, the resulting patch can contain multiple fields, and all field IDs within the patch are tracked and evaluated separately.

Since sugar-beet fields vary in size, the extracted patches differ in height, width, and number of temporal instances, but have a consistent number of channels. To standardize the spatial dimensions, a configurable patch size is used, and each extracted field is zero-padded to this size. Approximately 99.85% of the total sugar-beet fields lie within 40 hectares, which corresponds to 4000 pixels at a 10-meter resolution (as per data provided by Greenspin). Hence, we use a patch size of $(64, 64)$, covering 40.96 hectares, which is sufficient to encompass nearly all fields. In the cases where a field exceeds the target size, it is cropped to fit the configured patch dimensions. The output of this step are extracted sugar-beet fields as *Patch Tensors*, denoted as *Output 2*, with each patch tensor having the dimensions:

Output 2 Shape: $\mathbb{R}^{64\times64\times13\times T}$,
with $(64, 64)$ spatial size, 13 channels, and T number of temporal instances.

Note: We use the connected-components approach to preserve the spatial integrity of sugar beet fields, which is essential for applying convolutions later. Pixel selection via indexing would disrupt the spatial structure and result in an irregular number of pixels per field.

4.2.3 Fixing the temporal stack

The length of the temporal stack for each extracted sugar-beet field varies due to inconsistencies in the number of cloud-free temporal images. To standardize the temporal stack length, we select two cloud-free images per month for June, July, and August, and

one image from early September, corresponding to the sugar-beet growth cycle. Cloud masks are used to ensure only fully cloud-free images are selected, and partially or fully clouded temporal instances are discarded. A minimum time gap of five days is maintained between consecutive acquisitions. This process yields a consistent temporal stack of seven images for each patch. If fewer than seven cloud-free images are available for a patch, it is discarded.

While spatial or temporal interpolation could fill missing data, spatial interpolation risks smoothing localized stress signals, which are critical for this thesis. Temporal interpolation methods could be explored to retain more fields and increase stack length, but this is beyond the scope of the current work.

Finally, the border pixels of the sugar-beet fields are removed to avoid interference from surrounding vegetation or urban areas. The classical computer vision *binary_erosion* algorithm from *scipy.ndimage* [42] is used for this. The working of this function is detailed in algorithm 2. The output of this step (*Output 3*) is the modified patch tensor with a fixed number of temporal images. The dimension is as follows:

Output 3 Shape: $\mathbb{R}^{64\times64\times13\times7}$,
with $(64, 64)$ spatial size, 13 channels, and 7 temporal instances.

Data: binary_mask, input_image
Result: eroded_image: Eroded Image with border pixels removed
foreach *pixel in binary_image* **do**
  if *all neighbors are 1* **then**
    set pixel in input_image to 1;
  end
  else
    set pixel input_image to 0;
  end
end
return input_image as eroded_image;

Algorithm 2: Binary erosion algorithm

4.2.4 Converting patches to sub-patches

One of the main challenges of this thesis is the variability in sugar-beet field sizes. These fields vary significantly, ranging from 0.14 hectares to 52.44 hectares (numbers provided by Greenspin). To standardize the input, the extracted patches were padded to a consistent spatial dimension of $(64, 64)$ pixels in the previous steps. However, this padding introduces zero-value (black) pixels that contain no useful information and may introduce bias during the modeling process.

Sentinel-2 Band	Channel Type	B10	MVI	B4
Band 2	Blue	✓		✓
Band 3	Green	✓		
Band 4	Red	✓		✓
Band 5	Red Edge 1	✓		
Band 6	Red Edge 2	✓		
Band 7	Red Edge 3	✓		
Band 8	Near Infrared (NIR)	✓		✓
Band 8A	Narrow NIR (NNIR)	✓		
Band 11	Shortwave Infrared (SWIR 1)	✓		✓
Band 12	Shortwave Infrared (SWIR 2)	✓		
-	NDVI		✓	
-	EVI		✓	
-	MSI		✓	

Table 4.3: Matrix layout showing the use of Sentinel-2 bands and derived indices for tensors B10, MVI, and B4.

To minimize the impact of these zero-value pixels, we further divide the $(64, 64)$ patches into smaller $(4, 4)$ sub-patches. A sliding window of size $(4, 4)$ moves over the spatial dimensions of the patch tensor, extracting sub-patches across all channels and temporal instances. The spatial coordinates of each sub-patch, which correspond to locations in the original patch, are recorded for use during evaluation. Empty sub-patches, containing all zero-value pixels, are discarded. For sub-patches that are partially empty, zero-value pixels are replaced with the channel-wise average of the non-zero pixels within the sub-patch. This ensures that no zero-value pixels are passed to the model. The dimensions of the resulting *Sub-patch Tensor* is as follows:

Output 4 Shape: $\mathbb{R}^{4\times4\times13\times7}$,
with $(4, 4)$ spatial size, 13 channels, and 7 temporal instances.

4.2.5 Removing mask-channels and creating tensor variants

The final step of preprocessing is to remove irrelevant bands from the sub-patch tensors. Three tensor variants are created using different channel combinations for experimentation. Detailed information about the channel composition is provided below, and table 4.3 summarizes the corresponding channel configurations.

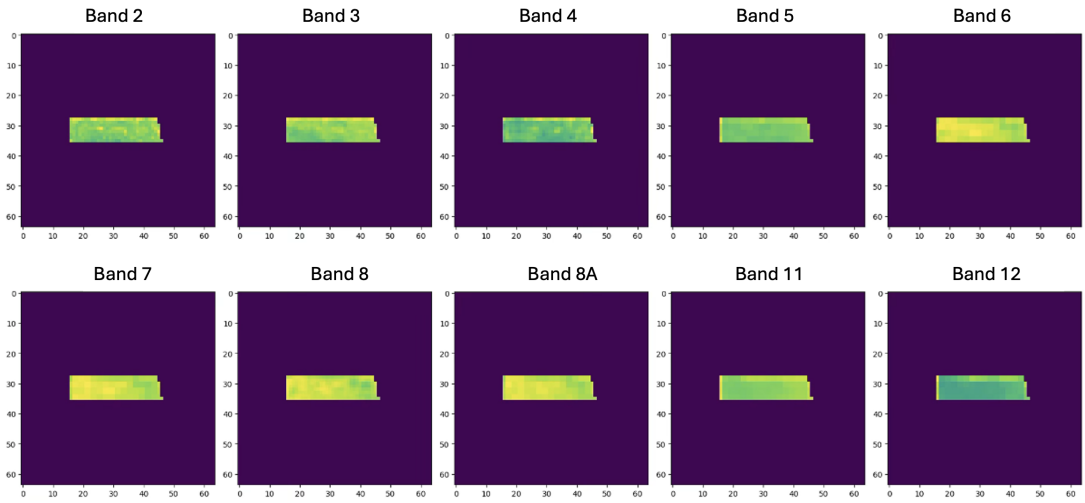


Figure 4.8: Sentinel-2 bands included in the B10 data tensor, shown for a sample field.

B10 tensor (all Sentinel-2 bands)

The Sentinel-2 images given by Greenspin consist of 10 Sentinel-2 bands. This tensor variant is composed of all these bands, and is denoted as B10. Specifically, the Sentinel-2 bands correspond to channel indices 0–9 as listed in table 4.1. Figure 4.8 shows the visualisation of these 10 bands for a sample sugar-beet field. Additional channels corresponding to masks (channels 10, 11, and 12) are excluded.

The B10 tensor has the dimensions $\mathbb{R}^{4\times4\times10\times7}$, with $(4, 4)$ spatial size, 10 channels, and 7 temporal instances.

MVI tensor (Multiple Vegetation Indices)

To evaluate the effect of using Vegetation Indices on model performance, a data tensor containing only vegetation indices as input channels is constructed. The indices Normalized Difference Vegetation Index (NDVI), Enhanced Vegetation Index (EVI), and Moisture Stress Index (MSI)—are computed pixel-wise using relevant spectral bands. Figure 4.9 visualises these Vegetation Indices for a sample sugar-beet field. The resulting Vegetation Indices are stacked along the channel dimension, while all original Sentinel-2 bands and mask channels are excluded from this tensor.

The resulting data tensor is referred to as MVI, and has the dimensions $\mathbb{R}^{4\times4\times3\times7}$, with $(4, 4)$ spatial size, 3 channels, and 7 temporal instances.

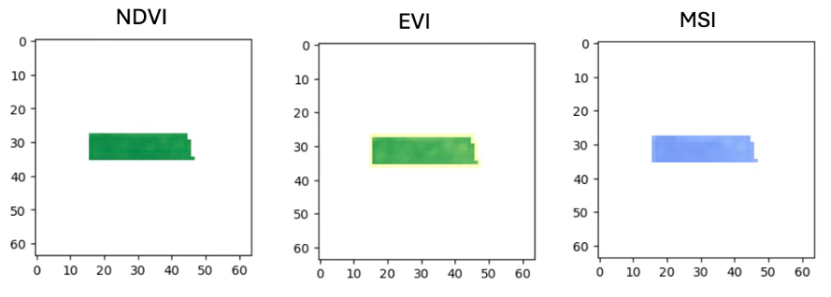


Figure 4.9: Vegetation Indices included in the data tensor MVI, shown for a sample field.

B4 tensor (4 Sentinel-2 bands)

For this variant, we use only the Sentinel-2 bands that are originally involved in the computation of the vegetation indices NDVI, EVI, and MSI. Vegetation Indices are derived from pixel values of image channels, and some information may be lost during this aggregation process. Therefore, we evaluate the effect of using the original spectral bands—Red, Blue, NIR, and SWIR1—that are used to compute these indices. Figure 4.10 illustrates these 4 bands for a sample sugar-beet field. All other Sentinel-2 bands and mask channels are excluded.

The resulting tensor is referred to as B4, and has the dimensions $\mathbb{R}^{4\times4\times4\times7}$, with $(4, 4)$ spatial size, 4 channels, and 7 temporal instances.

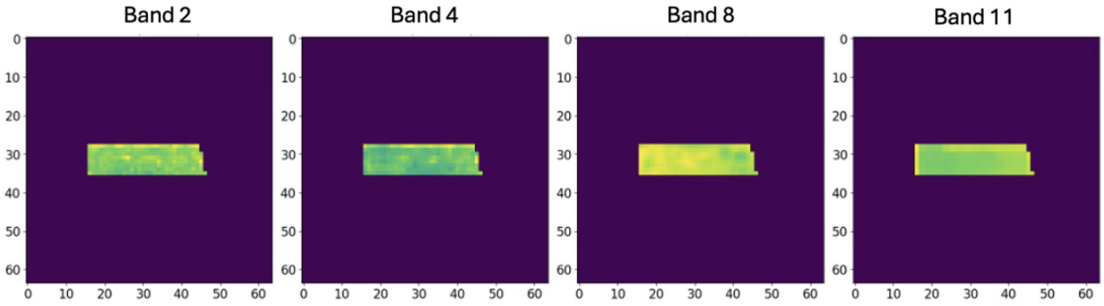


Figure 4.10: Sentinel-2 bands included in the data tensor B4, shown for a sample field.

Since the B10 tensor is used as default input for modeling, we denote the final preprocessed output as *Output 5 (B10)*.

Output 5 (B10) Shape: $\mathbb{R}^{4\times4\times10\times7}$,
with $(4, 4)$ spatial size, 10 channels, and 7 temporal instances.

5 Modeling

In this chapter, we begin by discussing temporal encodings and its implementation strategies. The following three sections outline the modeling process, which includes feature extraction, downstream clustering, and thresholding to obtain patch-level labels. Finally, we discuss the generation of localized stress maps as deliverables.

5.1 Temporal encodings

During data preprocessing, seven temporal instances are selected per image to ensure consistency across the dataset. These instances typically span the sugar-beet growth cycle, from June to early September. Throughout this period, the crop undergoes distinct developmental stages, and the reflectance values of image pixels change due to physiological growth as well as potential stress factors. Additionally, since acquisition dates vary across sugar-beet fields, the resulting temporal sequences are not uniformly spaced. Hence, knowing *when* each image was captured might provide critical information for distinguishing between healthy and stressed crops. To preserve this temporal context, we incorporate temporal encodings into the model inputs.

Inspired by the Transformer architecture [55], we use sinusoidal encodings to represent the acquisition dates. Each date is converted to a day-of-year value $d \in [1, 365]$, and the encoding is computed as:

$$\mathrm{e}_s = \sin\left(\frac{2\pi d}{365}\right), \quad \mathrm{e}_c = \cos\left(\frac{2\pi d}{365}\right)$$

This transformation maps the acquisition date to a continuous annual cycle. The calculated encodings are integrated in the input data during the forward pass. The loss is computed with respect to the original input tensor without encodings, ensuring that the encodings serve only as auxiliary information for the model. We integrate the encodings using two different strategies:

Strategy (A): Temporal encodings as extra channels

In this strategy, the sine and cosine temporal encodings are each broadcasted to create two additional channels. These encoding channels are appended to the input data tensor along the channel dimension. The resulting tensor, augmented with temporal information, is then passed through the model layers. Figure 5.1 provides a better understanding of the working of strategy A.

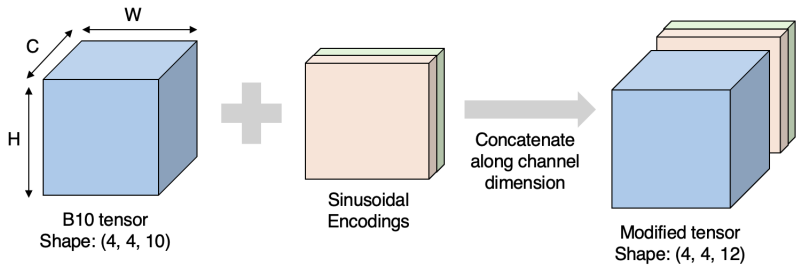


Figure 5.1: Strategy (A): Channel-wise concatenation of temporal encodings per temporal instance.

Strategy (B): Temporal encodings as element-wise addition:

In this strategy, rather than appending the temporal encodings as additional channels, we perform an element-wise addition of the encodings to the input tensor during the forward pass. The temporal encodings are broadcasted to match the dimensions of the original data tensor and then added element-wise. The resulting tensor, enriched with temporal context, is passed through the model layers. Figure 5.2 illustrates how the Strategy B operates.

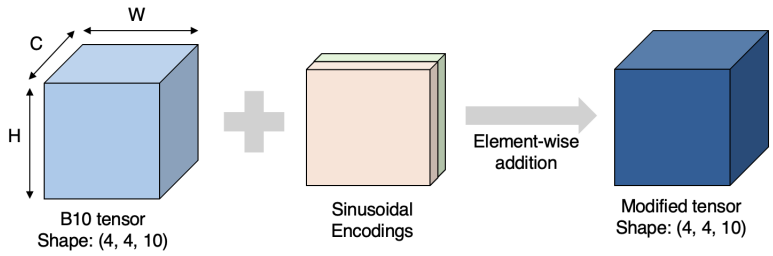


Figure 5.2: Strategy (B): Element-wise addition of temporal encodings per temporal instance.

As a preliminary step in the experiments, we evaluate these temporal encoding strategies and adopt the best performing approach for the autoencoder-based models throughout the thesis.

5.2 Feature extraction

In unsupervised learning, directly applying clustering algorithms to raw image data might lead to weak or noisy clusters, especially when working with complex inputs. Without any pre-processing, clustering may pick up irrelevant differences such as lighting conditions, noise, or surface artefacts, instead of actual vegetation trends. Hence, it is essential to transform the raw data into a more compact and informative representation. In this section, we discuss the feature extraction techniques that we use for the baselines and the proposed model.

5.2.1 Traditional methods

In this thesis, we include two classical approaches—histogram features and principal component analysis (PCA)—as baseline methods for feature extraction.

Histogram features

Each data sample for the B10 tensor is of shape $\mathbb{R}^{4\times4\times10\times7}$, representing a sequence of 7 time steps, each with spatial resolution (4,4) and 10 channels. To generate histogram features, we first flatten each sample into a 1D vector and compute a 34-bin histogram over the entire distribution of values. This choice is consistent with the widely used square-root choice heuristic [63], which suggests setting the number of bins k as

$$k = \lceil \sqrt{n} \rceil \ , \tag{5.1}$$

where n is the number of data points.

In our case, each sample has $n = 4\times4\times10\times7 = 1120$ values, resulting in $k = \lceil \sqrt{1120} \rceil \approx 34$. This bin count provides a compact yet expressive summary of the global distribution of pixel intensities across time and spectral channels.

The resulting histogram representations (obtained per data sample) are used subsequently for downstream clustering of sugar-beet sub-patches into stressed and healthy categories.

Principal Component Analysis

In the PCA baseline, we apply principal component analysis to reduce the channel dimensionality of each sample. Specifically, we retain the top 3 principal components across the 10-channel dimension for each time step. This choice captures the dominant spectral variance while maintaining a compact representation, resulting in a reduced tensor of shape $\mathbb{R}^{4\times4\times3\times7}$ with (4,4) spatial resolution, 3 channels and 7 temporal instances. We then flatten this tensor into a 1D feature vector to obtain a consistent input for clustering.

This PCA-based representation retains the channels with the most informative variance, while reducing the dimensionality. It offers a lightweight and interpretable baseline for comparison with the performance of more complex models, while also providing an insight into which Sentinel-2 spectral bands are most relevant for detecting stress in sugar-beets.

5.2.2 Autoencoder-based models

Beyond traditional methods, we explore 2D and 3D convolutions integrated with the autoencoder architecture. The models are trained in an unsupervised way, using the Mean Squared Error (MSE) loss (equation (3.4)). Once trained, the encoders are used to extract features by passing the preprocessed sub-patch tensors through them. The resulting latent representations serve as feature vectors for the downstream clustering task. The specific model architectures are detailed in the following sub-sections.

2D-convolutional autoencoder (2D_AE_B10)

The input to this model is of the form $\mathbb{R}^{H\times W\times(C\cdot T)}$, where after incorporating the temporal encodings, the temporal instances are stacked along the channel dimension. This architecture employs three 2D convolutional layers (Conv2d) that preserve the spatial dimensions, while progressively increasing the number of feature maps to effectively capture spectral patterns. The output of the convolutional layers is flattened, and passed through fully connected layers, which leads to the latent bottleneck $\mathbf{z} \in \mathbb{R}^d$. The decoder rebuilds the input tensor from the encoded representation, using transposed 2D convolutions.

We use the B10 tensor ($\mathbb{R}^{4\times4\times10\times7}$) as the default input and refer to this model as 2D_AE_B10, and use the features extracted from it as a baseline for comparison with other techniques.

3D-convolutional autoencoder (3D_AE_B10)

The input to this model is of the form $\mathbb{R}^{H\times W\times C\times T}$. Temporal encodings are first added to the input tensor, which is then passed through 3D convolutional layers that preserve the spatiotemporal dimensions of the data. The 3D convolutions operate over the spatial (height H, width W) and temporal (T) dimensions, allowing the model to jointly capture spatial and temporal patterns. Each convolutional layer uses an increasing number of feature maps to learn increasingly abstract representations, including spectral information across channels (C). The resulting tensor is flattened and passed through fully connected layers to produce a latent representation $\mathbf{z} \in \mathbb{R}^d$. The decoder reconstructs the input tensor from this latent representation using transposed 3D convolutions, restoring the original dimensions.

We use the B10 tensor ($\mathbb{R}^{4\times4\times10\times7}$) as the default input and refer to this model as 3D_AE_B10, which serves as the main proposed architecture in this thesis. Figure 5.3 shows the architecture for the 3D_AE_B10 model with temporal encodings.

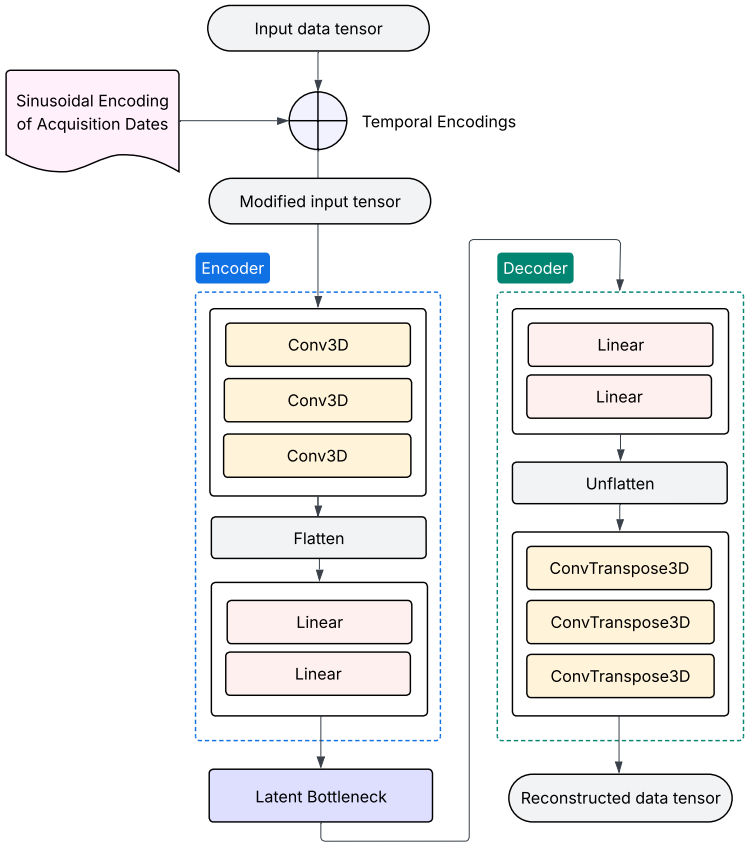


Figure 5.3: Architecture of 3D_AE_B10 with temporal encodings

5.3 Downstream clustering

In case of limited labels, machine learning techniques— specifically clustering is used to derive insights. Clustering is an unsupervised approach which groups similar data points based on intrinsic patterns and characteristics. There exist various clustering techniques [3], amongst which k-means is one of the most widely used due to its simplicity and efficiency.

In this thesis, we employ clustering due to the limited availability of ground-truth labels, making it a practical alternative to supervised classification in our problem setting. After feature extraction, we categorize the sugar-beet sub-patches as *Stressed* or *Healthy* by applying unsupervised clustering to the learned features $\mathbf{z}$. In the experiments section, we compare different clustering algorithms and select the most suitable one to use as the downstream clustering method throughout this thesis. The clustering methods are discussed in detail below.

5.3.1 K-means clustering

K-means clustering is a widely used unsupervised learning algorithm that partitions data into k non-overlapping clusters by minimizing intra-cluster variance. It begins by initializing k cluster centroids, then iteratively assigns each data point to its nearest centroid based on euclidean distance and updates the centroids as the mean of their assigned points. This process continues until the assignments no longer change significantly, indicating convergence. The algorithm aims to minimize the sum of squared distances between data points and their respective cluster centroids.

This is achieved by minimising the following objective function:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \ , \tag{5.2}$$

where $\mathbf{x}$ denotes a data point, $\boldsymbol{\mu}_i$ is the centroid of cluster C_i, and k is the total number of clusters.

While k-means is simple and efficient for large datasets, its performance can be affected by the initial placement of centroids, potentially leading to suboptimal solutions. Despite this, its speed and ease of implementation make it a popular choice for many clustering tasks.

5.3.2 K-medoids clustering

K-medoids clustering is an alternative to k-means that selects actual data points, called medoids as cluster centers rather than using the mean. This makes it more resilient to noise and outliers, as medoids are less affected by extreme values. The algorithm minimizes the sum of pairwise dissimilarities between points and their respective medoids, updating medoids iteratively to reduce intra-cluster distances.

The objective function of k-medoids can be written as:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} D(\mathbf{x}, \mathbf{m}_i) \ , \tag{5.3}$$

where $\mathbf{x}$ denotes a data point, $\mathbf{m}_i$ is the medoid of cluster C_i, and k is the total number of clusters.

Here, $D(\mathbf{x}, \mathbf{m}_i)$ represents the distance between data point $\mathbf{x}$ and the medoid $\mathbf{m}_i$. The medoid $\mathbf{m}_i$ is selected as the data point within cluster C_i that minimizes the sum of distances to all other points in the cluster. The distance D can be measured using various metrics such as euclidean, manhattan, or other distance measures.

K-medoids selects actual data points as cluster centers (medoids) instead of calculating mean values. However, it can be more computationally expensive, especially for large datasets, due to the repeated calculation of pairwise distances during medoid updates.

5.3.3 Agglomerative clustering

Agglomerative clustering is a hierarchical, bottom-up clustering method where each data point initially forms its own cluster. Clusters are then iteratively merged based on a linkage criterion—such as single linkage (minimum distance between clusters), complete linkage (maximum distance), or average linkage (average distance)—until the desired number of clusters is achieved. This process produces a dendrogram that visually represents the nested cluster structure, allowing flexible choice of cluster granularity.

The choice of linkage criterion influences how distances between clusters are computed and thus may affect the shape and quality of the resulting clusters. Although agglomerative clustering uncovers hierarchical relationships in data, it can be computationally expensive for large datasets due to repeated calculation and updating of distances during cluster merging.

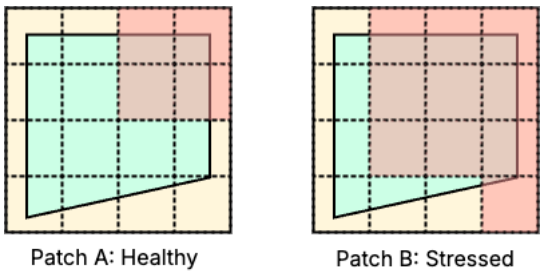


Figure 5.4: Thresholding example. Effect of stressed sub-patch count on patch-level label assignment.

5.4 Thresholding to obtain patch-level labels

Ground truth labels are provided at the patch level, with each sugar-beet field labeled as either *Healthy* or *Stressed*. However, one of the objectives of this thesis is to identify localized stress regions within individual fields. To facilitate this, each field is divided into $(4, 4)$ sub-patches, which are then individually modeled and clustered. This also enables the removal of sub-patches that do not contain any sugar-beet field pixels, before passing them to the model.

To evaluate the model performance against the ground truth, we aggregate the sub-patch predictions to produce a single prediction for the entire sugar-beet field. This is achieved using a simple thresholding approach: a field is classified as *Stressed* if the proportion of sub-patches identified as *Stressed* exceeds a predefined threshold value $\alpha \in [0, 1]$. Otherwise, the field is labeled as *Healthy*.

Formally, let N be the total number of sub-patches in a field, and let n_s denote the number of sub-patches predicted as *Stressed*. The field label P is determined as:

$$P = \begin{cases} 1, & \text{if } \frac{n_s}{N} > \alpha \\ 0, & \text{otherwise} \end{cases},$$

where $P = 1$ indicates a *Stressed* field, $P = 0$ indicates a *Healthy* field.

Example: Consider a threshold value of $\alpha = 0.5$. Figure 5.4 shows a simple 2D illustration of two field patches, comprising of 16 sub-patches each. Each patch consists of a different number of sub-patches clustered as *Stressed*. In case of Patch A, only 4 out of 16 sub-patches are identified as *Stressed*, resulting in a stressed proportion of 0.25. Since this is below the threshold, the patch is labeled as *Healthy*. In contrast, Patch B has 10 out of 16 sub-patches clustered as *Stressed*, corresponding to 0.625. As this exceeds the threshold, Patch B is labeled as *Stressed*.

Using a configurable threshold provides flexibility in defining the severity of stress re-

quired, to categorize an entire field as *Stressed*. This flexibility is useful in different scenarios—for example, a lower threshold can be set to detect fields with minimal stress for closer monitoring, while a higher threshold helps avoid false alarms by only marking fields with significant stress. It is also valuable during extreme environmental events, such as heavy rainfall or drought, which can significantly influence stress levels across the field. We use a default threshold of $\alpha = 0.5$ for modeling and experiments.

5.5 Generation of localized stress-maps

Apart from the sugar-beet field predictions, we also generate visual deliverables in the form of field-level stress maps. These stress maps are RGB images with localized stress regions highlighted using bounding boxes drawn over predicted stressed sub-patches. Since the default sub-patch size is 4, each bounding box corresponds to a (4, 4) pixel region in the original image space.

These visualizations are generated for both unlabeled fields and the labeled evaluation set, making them a valuable output for analysis. As more labeled data becomes available, the stress detection system can be further refined using a downstream classification task, making the visual outputs even more accurate and directly actionable for farmers. Figure 5.5 presents representative stress maps from the 2019 dataset.

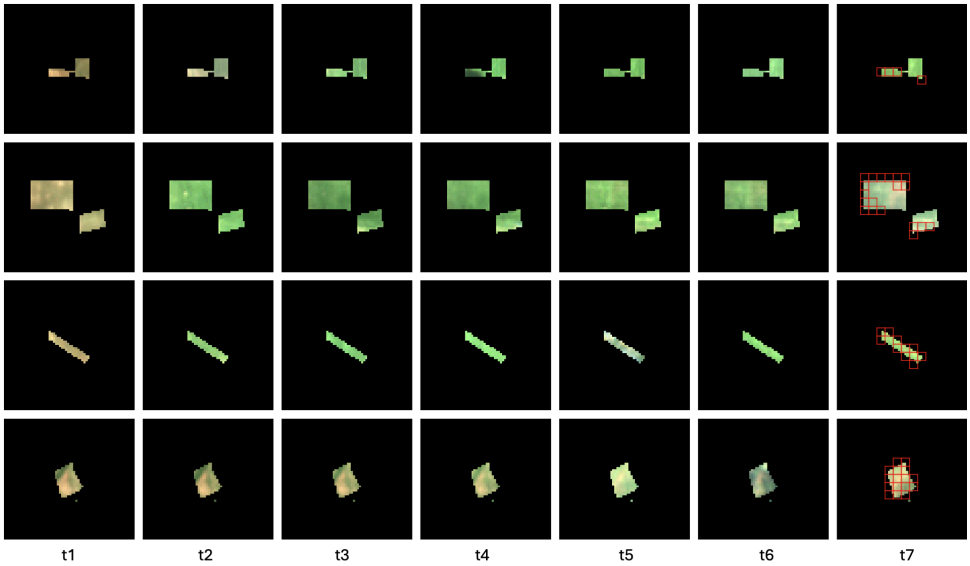


Figure 5.5: Localized stress maps for sugar-beet fields as deliverables, marked on the last temporal instance. Temporal instances are denoted by t1-t7.

6 Results and Experiments

In this chapter, we begin by outlining the experimental setup. We then present the preliminary tasks conducted to select a suitable downstream clustering algorithm and an effective temporal encoding strategy for the proposed model. This is followed by the main results and discussion, where we compare the performance of baseline methods with the main architecture. Additionally, we examine how variations in the input data and the sub-patch-to-patch threshold influences the overall performance, followed by stress detection on fields from the 2024 season.

6.1 Experimental setup

A total of 1857 unlabeled cloud-free sugar-beet fields corresponding to 1228 patches are partitioned into 33128 sub-patches. Of these, 80% are used for training and 20% for testing the autoencoder models (2D_AE_B10 and 3D_AE_B10). The latent features extracted from all subpatches of the 1857 unlabeled fields are subsequently used for downstream clustering. The evaluation set comprises of 61 cloud-free sugar-beet fields corresponding to 48 patches, further divided into 1197 sub-patches.

By default, the B10 sub-patch tensor is used for modeling, and a threshold of $\alpha = 0.5$ is applied to convert sub-patch predictions into patch-level labels. For training the autoencoder models, 2D_AE_B10 and 3D_AE_B10, the hyperparameters used are summarized in table 6.1. To ensure robust hyperparameter selection, we performed five-fold cross-validation on the unlabeled dataset while experimenting with different hyperparameter configurations for the autoencoders. The average reconstruction error on the validation split (over five folds) is used as the criterion for selecting the best-performing parameters.

Model performance is calculated over sugar-beet fields, specifically across the 61 fields in the evaluation set, using four metrics: accuracy, precision, recall and F1-score (reported in percentages). Accuracy measures the overall correctness of predictions, but it can be misleading in the presence of class imbalance, as noted in the data section. Recall captures the ability to correctly identify all stressed fields, but it may come at the cost of over-predicting stress, leading to false positives. Precision addresses this by measuring

Hyperparameter	Value
Latent size	32
Number of epochs	50
Loss function	MSE
Optimizer	Adam
Learning rate	0.001
Batch size	64

Table 6.1: Training hyperparameters used for the autoencoder-based models

how many predicted stressed fields are actually correct, helping reduce unnecessary false alarms. Together, precision and recall provide a more nuanced understanding of model performance. The F1-score balances these two metrics, making it particularly useful when the goal is to detect stressed fields reliably. However, it is important to examine which individual metric—precision or recall—contributes more to the F1-score for each model, to better interpret the results. The model performance is reported as the average over three independent executions, accounting for randomness in model initialization and training dynamics. All trained models are saved locally to support reproducibility.

6.2 Preliminary tasks

In this section, we perform two preliminary tasks: selecting the downstream clustering algorithm and choosing the temporal encoding strategy to be used with the autoencoder-based models.

6.2.1 Selecting a downstream clustering algorithm

We evaluate various clustering methods on the preprocessed sub-patch tensors and compare their performance. The best-performing method is selected as the downstream clustering algorithm for all subsequent experiments in this thesis. Specifically, we apply k-means, k-medoids, and agglomerative clustering to the B10 sub-patch tensor, using the default threshold ($\alpha = 0.5$) to derive patch-level labels.

It can be seen from table 6.2 that k-means clustering achieves the highest F1-score of 74.42%. This result is influenced by a high recall, indicating that k-means successfully identified most of the stressed fields. However, its lower precision reflects a tendency to misclassify healthy fields as stressed. Nevertheless, the resulting high F1-score highlights its effectiveness in capturing stressed cases, which is valuable in scenarios where detecting all of the stressed fields is critical.

Algorithm	Data Tensor	Accuracy	Precision	Recall	F1-score
k-means	B10	63.93	62.75	**91.43**	**74.42**
k-medoids	B10	**67.21**	**85.71**	51.53	64.29
Agglomerative	B10	42.62	50.00	54.29	52.05

Table 6.2: Comparison of clustering algorithms.

K-medoids clustering achieves the highest accuracy and precision, indicating it was better at minimizing false positives. However, it suffers from a significantly lower recall, meaning nearly half of the stressed fields were clustered wrongly as healthy. This drop in recall reduces its F1-score to 64.29%, making it less suitable in scenarios where it is important that stressed crops are not overlooked. Agglomerative clustering shows the lowest F1-score, with relatively poor precision and recall. This is likely due to its hierarchical approach, which appears less effective at modeling the complex spatiotemporal patterns present in the data.

The objective of this thesis is to detect as many stressed fields as possible while minimizing the false positives, especially given the potential imbalance in the data. The higher recall (and consequently higher F1-score) of k-means may be attributed to its use of centroids, which are not constrained to actual data points. In high-dimensional data such as the Sentinel-2 data used in this study, the true cluster centers might not coincide with specific data points. This flexibility enables k-means to form tighter and more balanced clusters than k-medoids. Furthermore, in our experiments, k-medoids required 20.3 minutes to execute, whereas k-means was completed in 0.1 seconds. Given this substantial difference in computational efficiency, and the fact that k-means provides the best trade-off between recall and precision, we select k-means as the downstream clustering method for subsequent experiments.

6.2.2 Selecting a temporal encoding strategy

As discussed in the modeling section, we experiment with two strategies for integrating temporal encodings into the model input. In Strategy A, the encodings are concatenated along the channel dimension, resulting in two additional input channels (figure 5.1). In Strategy B, the encodings are added element-wise to the input tensor, preserving the original shape (figure 5.2). Both strategies are evaluated using the B10 tensor, a default threshold of $\alpha = 0.5$, and the 3D_AE_B10 model, with the temporal encodings incorporated during the forward pass.

Table 6.3 shows the results of the two temporal encoding strategies. Both approaches demonstrate nearly equivalent performance, with very similar overall accuracy and F1-

Temporal Encoding	Data Tensor	Accuracy	Precision	Recall	F1-score
Strategy A	B10	**69.94**	**71.58**	79.04	75.09
Strategy B	B10	69.39	70.35	**80.95**	**75.21**

Table 6.3: K-means clustering results using various temporal encoding methods and model 3D_AE_B10.

scores. The element-wise addition method slightly outperforms the extra channel method in terms of F1-score, driven by its higher recall. This suggests that element-wise addition may help the model capture temporal patterns in a way that slightly enhances its sensitivity to stressed fields. On the other hand, the extra channel method yields a marginally higher precision, indicating it produces slightly fewer false positives.

However, it is worth noting that including temporal encodings as separate input channels increases the input dimensionality, resulting in slightly higher computational cost during training. In this case, the Strategy A (extra channels) required approximately 373 seconds of training time, compared to 354 seconds for Strategy B (element-wise addition).

Overall, the results suggest that both encoding strategies are viable, with element-wise addition offering a slight advantage when recall and F1-score is prioritized, while also being a computationally efficient alternative. Hence, for the model comparisons and all subsequent experiments, we adopt temporal encoding Strategy B (element-wise addition) for the autoencoder models (2D_AE_B10 and 3D_AE_B10).

6.3 Results and discussion

In this section, we evaluate the performance of the proposed architecture against several baseline methods. We also investigate how variations in input data and sub-patch-to-patch threshold affects the stress detection performance. Specifically, we test different input representations by varying the channel compositions and sub-patch sizes. We also assess the influence of the sub-patch-to-patch threshold α on the clustering results. Finally, we validate the full stress detection pipeline on an independent dataset from the 2024 season, covering the entire workflow from preprocessing to clustering evaluation.

Method	Temporal Encoding	Data Tensor	Accuracy	Precision	Recall	F1-score
Raw Data	✗	B10	63.93	62.75	**91.43**	74.42
Histogram features	✗	B10	52.46	55.77	82.86	66.67
PCA	✗	B10	57.38	59.18	82.86	69.05
2D_AE_B10	✗	B10	60.11	63.96	73.34	67.78
3D_AE_B10	✗	B10	59.56	62.80	72.38	67.25
2D_AE_B10	✓	B10	59.01	60.07	84.76	70.26
3D_AE_B10	✓	B10	**69.40**	**70.36**	80.95	**75.21**

Table 6.4: K-means clustering results on sugar-beet fields using various feature extraction methods.

6.3.1 Evaluating the proposed model against baselines

The proposed 3D_AE_B10 model with temporal encodings is evaluated against four baselines for feature extraction. The first baseline applies k-means directly on the sub-patch tensors without any feature abstraction. The second baseline performs k-means on histogram features computed per data sample. The third baseline uses PCA to reduce the number of channels per sample from 10 to 3 by selecting the top 3 principal components. The resulting tensor is then flattened for each sample and used for k-means clustering. Additionally, the fourth comparison is with the 2D_AE_B10 model with temporal encodings, which employs 2D convolutions. We also evaluate both 2D_AE_B10 and 3D_AE_B10 models without temporal encodings to assess the impact of explicitly incorporating the temporal information. Table 6.4 shows the comparison of downstream clustering results using various feature extraction methods, with a focus on the effect of temporal encodings on model performance. We use the B10 sub-patch tensor as input and the default sub-patch-to-patch threshold ($\alpha = 0.5$).

The 3D_AE_B10 model with temporal encodings achieves a high F1-score of 75.21%, demonstrating a strong balance between recall and precision—both of which are crucial in and detecting all stressed fields and minimizing false positives. In addition, it attains the highest accuracy across all methods, indicating reliable overall clustering performance. The 3D convolutional architecture leverages the spatio-temporal-spectral information in the data, enabling it to more accurately capture subtle temporal patterns associated with the stress progression. In contrast, the 2D_AE_B10 with temporal encodings shows a higher recall (84.76%) compared to 3D_AE_B10 with temporal encodings, suggesting it is better at detecting stressed fields. However, its lower precision might lead to more false positives, which is reflected in a lower F1-score. This indicates that while temporal encoding benefits both autoencoder architectures, the the ability of 3D convolutions to jointly analyze the spatial and temporal features provides a more robust representation for stress detection.

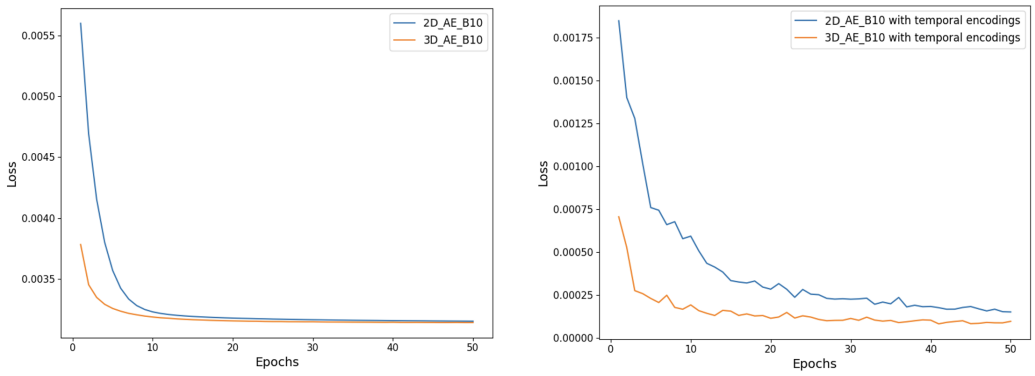


Figure 6.1: Test reconstruction losses for 2D_AE_B10 and 3D_AE_B10, with and without temporal encodings.

When temporal encodings are removed, the 3D_AE_B10 model exhibits a noticeable drop in performance, as reflected in reductions across all four metrics. The 2D_AE_B10 model also experiences a significant drop in recall, which is however offset by a slightly higher precision, resulting in only a marginal decrease in F1-score. This highlights the importance of incorporating temporal information, especially when used in combination with 3D convolutions. Without temporal information, the models might struggle to learn meaningful representations of the underlying progression of stress, leading to less reliable clustering.

The test loss plots for the 2D and 3D autoencoder models further support these findings. As shown in figure 6.1, both models exhibit consistent convergence, but the 3D_AE_B10 with temporal encodings achieves a lower loss over the epochs. This suggests that including temporal encodings enables better reconstructions, and that the 3D model is slightly more effective than the 2D model at capturing and compressing spatiotemporal information. Figure 6.2 shows the ability of 3D_AE_B10 model with temporal encodings to reconstruct temporal reflectance patterns.

The raw data yields the highest recall among all methods, which contributes to a relatively high F1-score. This suggests that the unprocessed input might already contain strong discriminative information for identifying stress in sugar-beet fields. The F1-score, only marginally below than that of the 3D_AE_B10 model with temporal encodings, further highlights its effectiveness in detecting positive instances. However, the comparatively lower precision indicates a tendency to over-predict stress, leading to more false positives. This trade-off might affect the model reliability, as high recall alone may not be sufficient without a good precision, to ensure robust clustering.

The PCA and histogram-based baselines follow a similar trend. Both methods achieve relatively high recall, showing their ability to capture stressed samples. However, the lower precision values lead to moderate F1-scores, reflecting a tendency to over-predict

Channel Index	Sentinel-2 Band	Band Name
6	Band 8	NIR
8	Band 11	SWIR 1
5	Band 7	Red Edge 3

Table 6.5: Top 3 bands retained by PCA over the 2019 dataset, ranked by importance.

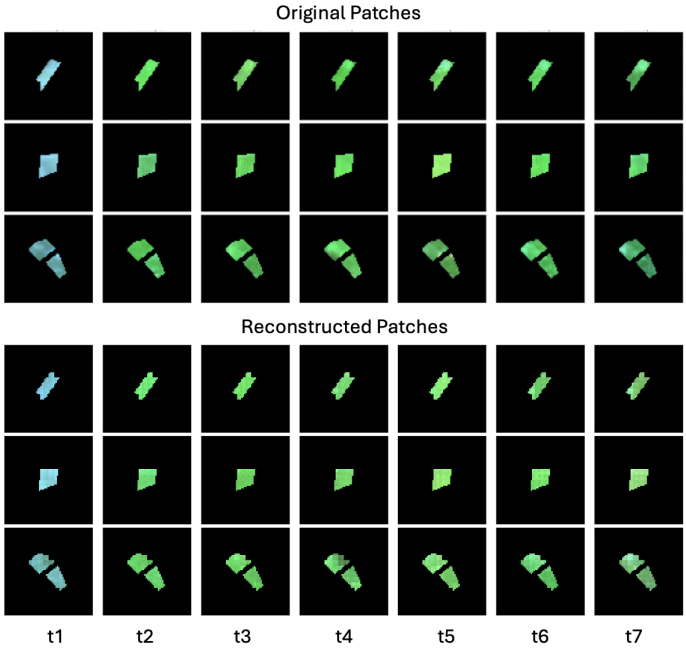


Figure 6.2: Original sugar-beet patches and their 3D_AE_B10 (with temporal encodings) sub-patch reconstructions mapped back onto the patches. Temporal instances are denoted by t1–t7.

stress. Nevertheless, their performance suggests that simple statistical and dimensionality reduction techniques still provide valuable signals for stress detection. Using PCA, we also identify the most frequently selected spectral bands across the 2019 dataset, as reported in table 6.5. These bands capture specific traits such as biomass, moisture content, and pigment variations, and their frequent selection suggests they play a significant role in detecting stressed fields.

In summary, while baseline models such as PCA and histogram-based features achieve high recall, they suffer from low precision, resulting in modest F1-scores and accuracies. Using raw data yields the highest recall and an F1-score comparable to the 3D_AE_B10 model with temporal encodings, indicating strong ability to capture stressed cases—but with a tendency to over-predict stress. The model 3D_AE_B10 with temporal encodings strikes an effective balance of precision and recall, and highlights the role of temporal encodings in combination with 3D convolutions for enhancing feature representations.

Method	Temporal Encoding	Data Tensor	Accuracy	Precision	Recall	F1-score
3D_AE_B10	✓	B10	**69.40**	**70.36**	**80.95**	**75.21**
3D_AE_MVI	✓	MVI	52.46	56.27	75.24	64.32
3D_AE_B4	✓	B4	60.65	62.81	76.19	68.84

Table 6.6: Clustering results based on features extracted using the 3D_AE architecture with temporal encodings, trained on input with varying channel composition.

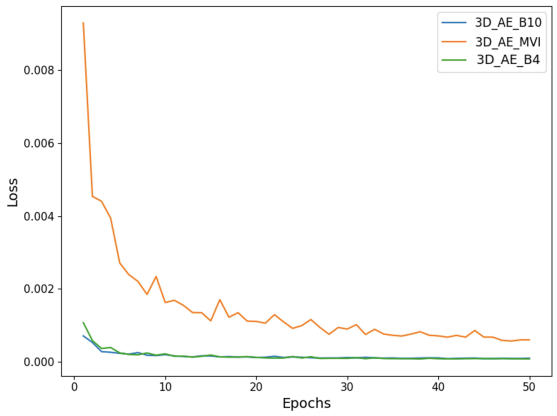


Figure 6.3: Test reconstruction losses for 3D_AE architecture with temporal encodings, using input with varying channel composition.

6.3.2 Experimenting with channel compositions

This study also explores the role of vegetation indices in stress detection. While B10 is the primary input for model comparison, we additionally evaluate the 3D_AE architecture with temporal encodings, using two alternative tensors: MVI including three different vegetation indices (NDVI, EVI and MSI), and B4 including the four Sentinel-2 bands used to compute MVI (Sentinel-2 bands 2, 4, 8 and 11). The band compositions are detailed in table 4.3. We use the default threshold of $\alpha = 0.5$, and denote the models based on their input tensor variant-3D_AE_MVI for MVI and 3D_AE_B4 for B4.

Table 6.6 presents the clustering performance of the 3D_AE model using different input data tensors. The B10 tensor yields the best overall performance, scoring highest across all four metrics. This suggests that richer spectral information enhances the ability of the model to learn meaningful features in an unsupervised setting. In contrast, the MVI tensor, based solely on vegetation indices, performs the worst across all metrics, indicating that pre-computed indices may lack the spectral depth required for effective representation learning. The B4 tensor (comprising 4 raw Sentinel-2 bands) performs better than MVI, demonstrating that even a limited number of raw spectral bands can capture more useful information than derived indices in this context.

Figure 6.3 illustrates the test reconstruction loss curves for models trained on different input tensors. While the differences in the losses are slight, the 3D_AE_B10 and 3D_AE_B4 models converge to the lowest loss, aligning with the clustering results. The 3D_AE_MVI model shows a slightly higher test loss, reflecting less stable reconstruction. Notably, the loss curve for 3D_AE_B4 closely tracks that of 3D_AE_B10, which further supports its effectiveness despite using fewer spectral bands.

6.3.3 Varying the sub-patch-to-patch threshold

We further examine the impact of varying the sub-patch-to-patch threshold $\alpha \in [0, 1]$. A sugar-beet field is labeled as stressed if the proportion of its sub-patches predicted as stressed exceeds the threshold α. Figure 6.4 presents the precision-recall and F1-score curves for the 3D_AE_B10 model with temporal encodings and the raw data baseline, with α varying from 0.1 to 1.0.

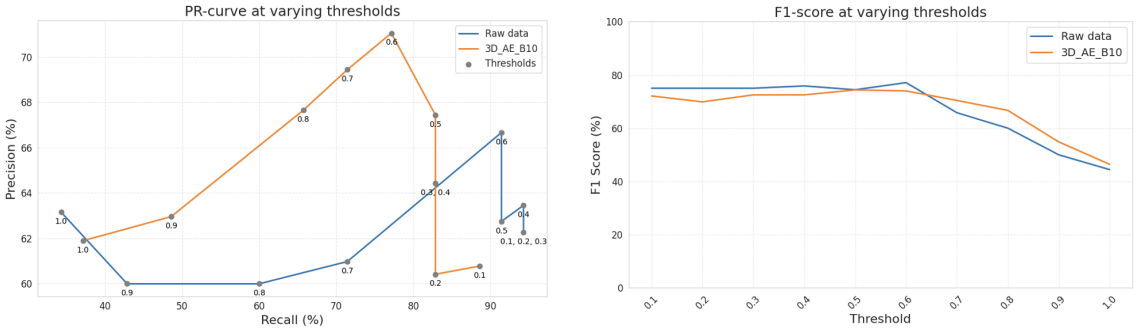


Figure 6.4: Precision-Recall curve for varying the sub-patch-to-patch threshold α.

At lower thresholds, the baseline achieves higher recall than 3D_AE_B10 with temporal encodings, and nearly equivalent F1-scores, highlighting its strength in detecting stressed fields. As the threshold increases, a larger proportion of sub-patches must be predicted as stressed for the entire field to be labeled as stressed. Under these stricter conditions, 3D_AE_B10 with temporal encodings begins to outperform the baseline, particularly in precision—and consequently in F1-score. This shift reflects the tendency of the baseline model to incorrectly label healthy fields as stressed when the threshold is high. While both models perform comparably around the default threshold $\alpha = 0.5$, 3D_AE_B10 with temporal encodings shows greater robustness at higher thresholds, maintaining better precision and a slightly higher F1-score. This suggests that the spatiotemporal representations learned by 3D_AE_B10 with temporal encodings lead to more reliable stress detection under higher thresholds.

The impact of varying α also highlights complementary strengths of the two models under different real-world scenarios. For example, in conditions where stress signals are

Method	Temporal Encoding	Sub-patch Size	Accuracy	Precision	Recall	F1-score
3D_AE_B10_4	✓	4	**69.40**	**70.36**	80.95	**75.21**
3D_AE_B10_8	✓	8	61.21	66.87	67.62	66.35
3D_AE_B10_16	✓	16	56.28	58.21	**84.76**	69.02

Table 6.7: Clustering results based on features extracted using the 3D_AE architecture with temporal encodings, trained on input with varying sub-patch size.

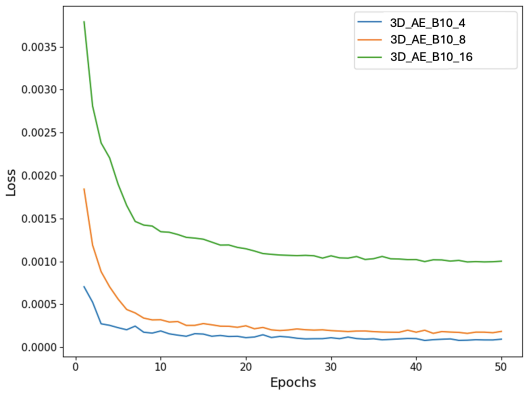


Figure 6.5: Reconstruction test loss for 3D_AE architecture with temporal encodings, using input with varying sub-patch size.

more subtle, such as during widespread drought or early-stage disease onset, a lower threshold may be desirable to flag even mildly affected fields. In such cases, the raw data baseline, with its tendency toward higher recall, might be more effective for broader stress detection. Conversely, in scenarios where severe and localized stress is actionable, such as after heavy rainfall or when prioritizing high-confidence interventions, a higher threshold is more appropriate. Under these stricter conditions, 3D_AE_B10 with temporal encodings might be more robust, offering higher precision and overall better reliability. This suggests that the two models may serve complementary roles depending on operational priorities and the environmental conditions.

6.3.4 Varying the sub-patch size

In this section, we explore the affect of varying the sub-patch size on model performance. We evaluate sub-patch sizes of 4, 8, and 16 using the 3D_AE architecture with temporal encodings, followed by downstream k-means clustering. We use the B10 tensor and sub-patch-to-patch threshold $\alpha = 0.5$ as default, and denote the models as 3D_AE_B10_4, 3D_AE_B10_8, and 3D_AE_B10_16 for sub-patch sizes 4, 8 and 16 respectively.

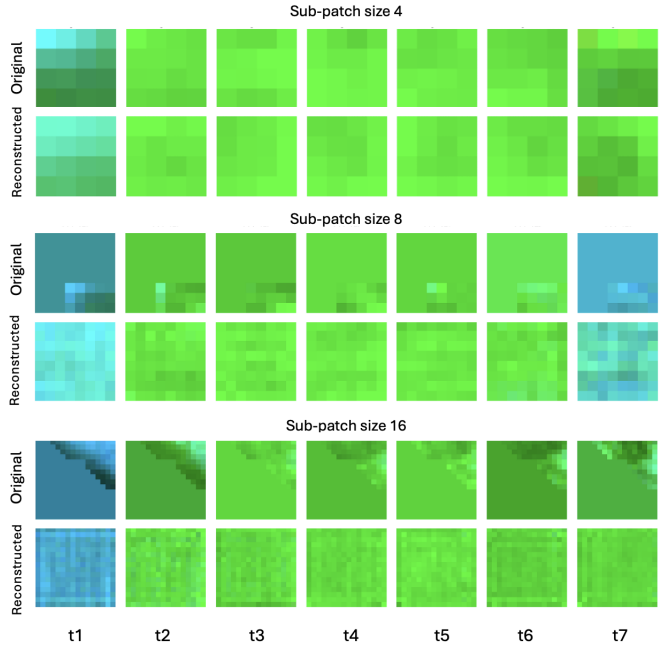


Figure 6.6: Original and reconstructed images for 3D_AE_B10 with temporal encodings, trained with varying sub-patch sizes. Temporal instances denoted by t1–t7.

The results shown in table 6.7 indicate that using smaller sub-patches leads to better performance across most evaluation metrics. The 3D_AE_B10_4 model achieves the overall best results with the highest F1-score, suggesting that finer spatial granularity helps to capture the local features better. In contrast, while the 3D_AE_B10_16 model achieves the highest recall, this comes at the expense of low precision and accuracy. The elevated recall may be explained by the increased number of padded black pixels in the larger sub-patches, which, when replaced with the mean during preprocessing, result in more uniform input regions. This uniformity might weaken the real stress signals and cause the model to wrongly focus on these smooth areas. As a result, the model may wrongly identify these uniform patches as stressed, causing false positives. This bias might increase the recall but it lowers the precision, which reduces the reliability.

Figure 6.5 illustrates the test reconstruction loss across the three sub-patch configurations. The model trained with a sub-patch size of 4 consistently achieves the lowest loss, indicating better reconstruction quality and more effective feature learning. As the sub-patch size increases, the reconstruction loss is higher—moderately for the 3D_AE_B10_8 configuration and more significantly for the 3D_AE_B10_16 model. This trend corresponds with the observed decline in clustering performance for larger sub-patch sizes.

Figure 6.6 visualizes the reconstruction outputs across different sub-patch sizes. The 3D_AE_B10_4 model achieves clean reconstructions, preserving the subtle spatial variations with minimal noise. In contrast, reconstructions from the 3D_AE_B10_8 model

show increased pixelation and some loss of local detail. The 3D_AE_B10_16 outputs exhibit the most degraded reconstructions, with significant smoothing in regions that originally contained higher variability. These observations align with the clustering metrics, reaffirming that smaller sub-patches enable more precise feature representations.

6.3.5 Validating the stress detection system on 2024 data

Our developed stress detection system is implemented as a single, modular script that loads raw Sentinel-2 data, performs preprocessing to generate B10 sub-patch tensors, trains the model 3D_AE_B10 with temporal encodings, and outputs field-level predictions as well as localized stress maps for sugar-beet fields. The system is fully functional and requires only path specifications in the configuration file. We applied this system to data from the 2024 sugar-beet season. Due to unusually high rainfall during the year, the dataset lacked sufficient cloud-free images to meet the default requirement of seven observations per field. As a result, we used four temporal instances—one each from June, July, August, and September—spanning the growth cycle. The config.py file can be adjusted directly to accommodate different numbers of temporal instances.

Experimental setup for 2024 season

A total of 2026 unlabeled, cloud-free fields from the 2024 season corresponding to 1491 patches, are partitioned into 50529 sub-patches. Of these, 80% are used for training and 20% for testing the model 3D_AE_B10 with temporal encodings. Latent features extracted are subsequently used for downstream clustering via the k-means algorithm. For evaluation, a separate set of 40 cloud-free fields corresponding to 33 image patches is used, further divided into 1213 sub-patches. Among these, 35 fields are labeled as stressed and 5 as healthy, resulting in an imbalanced evaluation set. The B10 sub-patch tensors serve as the input for modeling, and the hyperparameters are adopted from the configuration tuned during modeling the 2019 data. To obtain patch-level labels, a threshold of $\alpha = 0.5$ is applied. Model performance is assessed at the field level across the 40 evaluation fields, using four standard metrics: accuracy, precision, recall, and f1-score. Reported scores are averages over three independent runs, to account for randomness in model initialization and training dynamics.

Results and discussion

The evaluation results presented in table 6.8 reflect the performance of the proposed stress detection pipeline on sugar-beet data from the 2024 season, under the constrained

Metric	Score (%)
Accuracy	77.50
Precision	86.09
Recall	88.57
F1-score	87.28

Table 6.8: Clustering results on 2024 data using 3D_AE_B10 with temporal encodings.

temporal setup. Despite the reduction from seven to four temporal observations per field due to cloud cover, the model achieves strong clustering results. The high F1-score highlights the ability of the model ability to maintain a robust balance between precision and recall—particularly important in this setting, where the ground truth is highly imbalanced. Notably, the model exhibits a high recall, indicating its effectiveness in identifying most of the stressed instances. This sensitivity is critical in stress detection, where false negatives (i.e., missed stress cases) could have significant implications. Meanwhile, the precision score suggests that the system avoids over-flagging stress, thus minimizing false alarms and unnecessary intervention.

While the accuracy is 77.50%, it must be interpreted with caution due to the pronounced class imbalance. In such cases, accuracy alone can be misleading, as models skewed toward the majority class (stressed) may still yield deceptively high values. However, the strong F1-score, combined with high precision and recall, indicates that the model is not overfitting to the dominant class and successfully differentiates between both stressed and healthy categories.

Despite being trained on a reduced number of temporal instances, the model demonstrates robustness in capturing discriminative spectral-temporal patterns from B10 subpatch tensors. This performance illustrates the effectiveness of the temporal encodings and supports the system's adaptability to real-world settings, where temporal sequences might often be incomplete due to atmospheric constraints.

Although the evaluation set is relatively small, and thus limits the statistical strength of the results, the findings still offer encouraging evidence that the proposed system generalizes well across different growing seasons and field conditions.

Comparison with 2019 results

The comparative results between the 2019 and 2024 seasons are illustrated in figure 6.7. The 2024 sugar-beet season was characterized by prolonged heavy rainfall and widespread disease outbreaks, which significantly impacted the field conditions. These factors not only made the stress symptoms intense across the landscape, but also re-

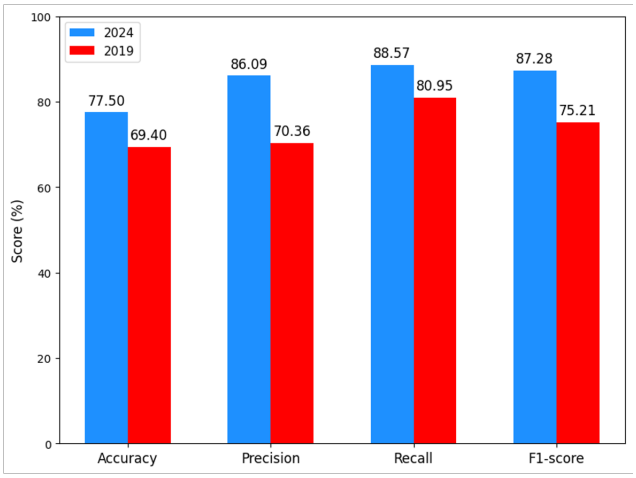


Figure 6.7: Comparison of clustering results from 2019 and 2024 sugar-beet season, using 3D_AE_B10 with temporal encodings.

duced the availability of cloud-free images, limiting the temporal resolution to four key observations instead of seven. Additionally, the 2024 evaluation set is imbalanced, with a majority of fields labeled as stressed. Despite these challenges, the system successfully identified stressed fields with high reliability. This may be attributed to the severity and consistency of stress signals during the 2024 season, as also noted by Greenspin, which likely provided clearer distinctions in the spectral data for the model to learn from. This also indicates that the model was able to extract meaningful spectral-temporal patterns even from limited temporal instances, which is particularly valuable in scenarios where acquisition gaps are present due to climatic conditions such as rain.

In contrast, the 2019 season provided a more balanced evaluation set, with typical climatic conditions. The slightly larger and more evenly distributed evaluation set allowed the model to perform in a relatively neutral environment. This less challenging environment of 2019 may have led the model to extract more generic representations. While the model performed well, its predictions are comparatively conservative. This is likely due to possibility of the stress signals being subtle, making the stressed and healthy categories less distinct.

This experiment highlights two key insights: first, the developed system demonstrates strong generalization across varying field conditions; and second, even with reduced temporal observations, the system is capable of learning sufficiently rich representations to achieve modest performance.

Figure 6.8 shows the reconstructed outputs of the 3D_AE_B10 model with temporal encodings for the 2024 dataset, while figure 6.9 presents the final deliverable stress maps, highlighting localized stressed regions in sugar-beet fields.

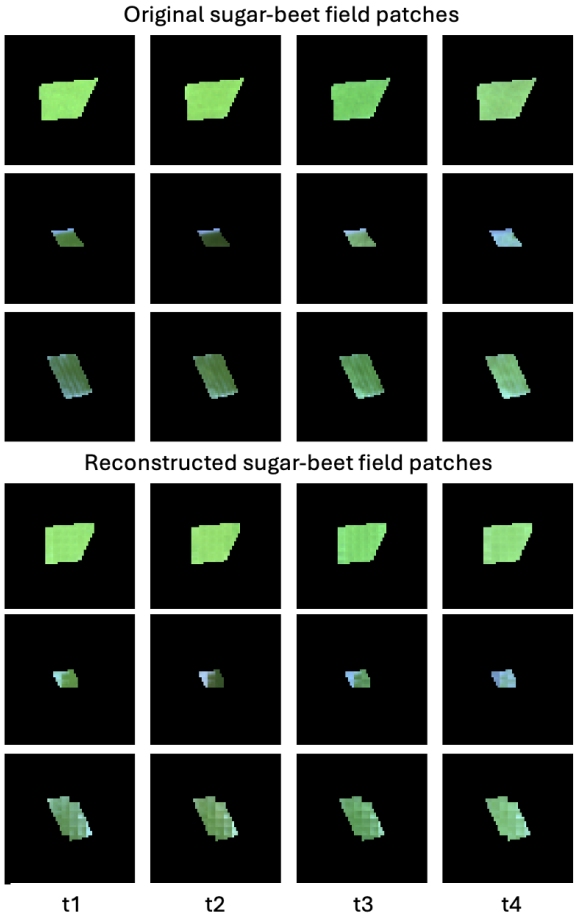


Figure 6.8: Original 2024 sugar-beet patches and their 3D_AE_B10 (with temporal encodings) sub-patch reconstructions mapped back onto the patches. Temporal instances are denoted by t1–t4.

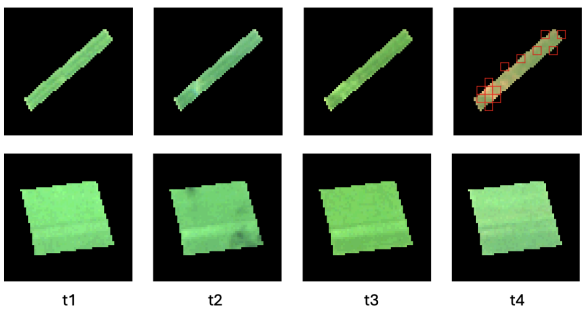


Figure 6.9: Localized stress maps for 2024 sugar-beet fields as deliverables, marked on the last temporal instance. Temporal instances are denoted by t1-t4.

7 Intermezzo: Masked Autoencoders for Satellite Images

In this section, we examine the architecture of SatMAE (Satellite Masked Autoencoders)[11], a method that extends the Masked Autoencoder (MAE) framework[21] to satellite imagery for representation learning. We explore this approach because it is tailored for Sentinel-2 data, and leverages representation learning for extracting meaningful spectral and temporal features through its architecture variants—spectral-SatMAE and temporal-SatMAE respectively.

As part of this thesis, we conducted experiments using the temporal-SatMAE architecture to obtain latent feature representations from temporal data for downstream clustering. However, the approach did not yield satisfactory results in terms of reconstruction quality. Despite the lack of empirical success, these experiments provided valuable insights into the applicability and limitations of the SatMAE framework in our specific setting. Accordingly, we have included the implementation in the thesis codebase for reproducibility and future reference.

This section briefly introduces the concept of masked autoencoders, outlines the experimental setup, highlights the challenges encountered, and discusses possible reasons for the observed performance limitations.

7.1 Introduction to SatMAE

SatMAE (Satellite Masked Autoencoders) is a self-supervised learning framework designed to effectively handle the unique properties of satellite imagery, especially its temporal and multi-spectral characteristics. It builds upon the foundational Masked Autoencoder (MAE) architecture, and introduces key modifications to create two different architectures as follows:

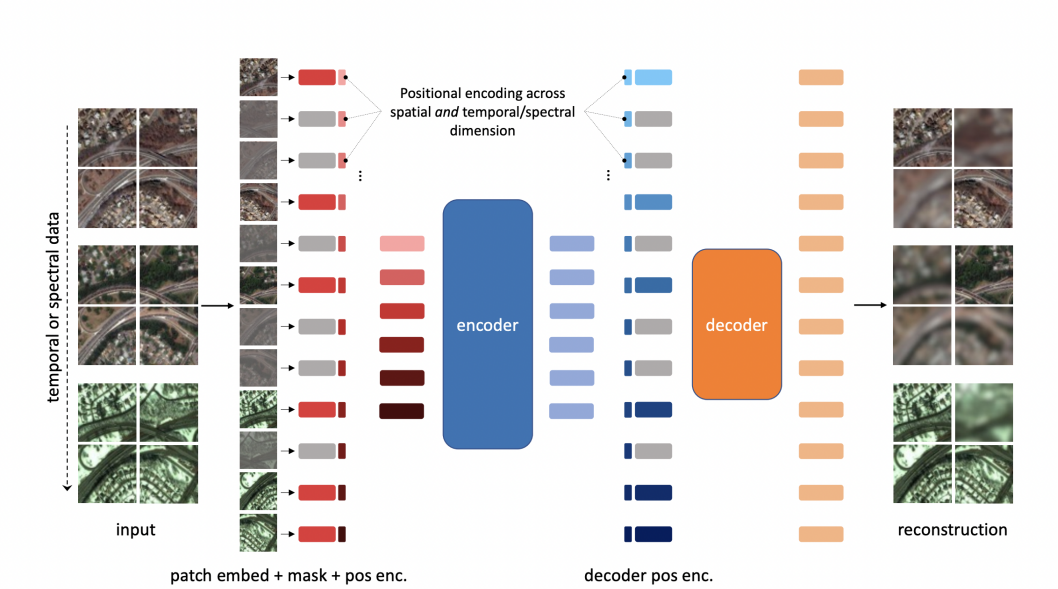


Figure 7.1: Temporal-SatMAE Architecture

Temporal SatMAE

Unlike conventional MAEs that mask spatial patches independently, SatMAE performs masking along the temporal dimension of satellite image time-series. Specifically, it masks a subset of patches across different time steps, enabling the model to reconstruct missing temporal observations by leveraging context from other temporal instances. This approach capitalizes on the inherent temporal redundancy and correlations in satellite data. To explicitly incorporate temporal order and dynamics, SatMAE adds temporal positional embeddings to each input. These embeddings help the encoder to understand the chronological sequence of observations and facilitate learning of temporal patterns such as crop growth cycles or phenological changes.

Spectral SatMAE

Satellite sensors capture data across multiple spectral bands (e.g., visible, near-infrared, shortwave infrared), each sensitive to different surface properties. SatMAE groups these spectral bands and assigns distinct positional encodings for each group, allowing the encoder to differentiate between spectral dimensions. This spectral positional encoding enables the model to learn cross-band relationships and better represent spectral signatures that are critical for tasks like vegetation monitoring, stress detection, or land cover classification.

The overall architecture follows the MAE paradigm: the encoder processes only the visible (unmasked) patches, generating latent representations, while the decoder reconstructs the full set of patches including the masked ones, using learnable mask tokens. The training objective is to minimize the reconstruction loss (commonly MSE loss) between the original and reconstructed patches over the masked regions. Figure 7.1 shows the architecture for temporal-SatMAE, as provided in [11].

7.2 SatMAE for sugar-beet stress detection

We implement the temporal-SatMAE architecture using satellite images from the 2019 sugar-beet season. A total of 1857 unlabeled, cloud-free sugar-beet fields, corresponding to 1228 patches, are used with an 80:20 train-test split. Latent features extracted are subsequently used for downstream k-means clustering. Evaluation is conducted on a separate set of 61 cloud-free sugar-beet fields, corresponding to 48 patches. Unlike our primary data preprocessing pipeline, we do not divide each $(64, 64)$ patch into $(4, 4)$ sub-patches. Instead, the sub-patch creation is handled internally by the SatMAE architecture using the default *patchify* function. We set the sub-patch size to 16, as used in the original implementation. Moreover, we use only three temporal observations—from July, August, and September—which aligns with the temporal setup of the original SatMAE study. As the temporal-SatMAE operates only on RGB channels, the remaining Sentinel-2 spectral bands are discarded. Each input tensor thus has dimensions:

$$\mathbb{R}^{64\times64\times3\times3} ,$$

corresponding to $\mathbb{R}^{H\times W\times C\times T}$, where H, W, C and T represents the height, width, number of channels and number of temporal instances respectively.

The encoder consists of 6 ViT [16] blocks, and the decoder consists of 4 ViT blocks. We use fixed spatial positional encodings for sub-patches within the $(64, 64)$ patch, and sinusoidal temporal encodings based on the acquisition date (day, month, year) of each observation. A masking ratio of 75% is applied during training, and the latent size is set to 32. Training is performed with a batch size of 64, using the Adam optimizer with a learning rate of 0.0001 for 50 epochs. The model is trained to reconstruct the masked sub-patches using the Mean Squared Error (MSE) loss (equation (3.4)).

Result and discussion:

After training the temporal-SatMAE model, we observed that the reconstructed outputs were entirely black. This issue is likely attributed to the preprocessing step where sugar-beet fields of varying sizes were padded to a uniform shape of (64, 64) pixels. As noted earlier, approximately 99.85% of sugar-beet fields are smaller than 40 hectares. Table 7.1 presents the field size statistics for the 2019 season, as provided by Greenspin,

Statistic	**Field Size (hectares)**
Average	3.04
Median	2.19
1st Quartile	1.34
3rd Quartile	3.60
Minimum	0.14
Maximum	52.44
Standard Deviation	3.29

Table 7.1: Sugar-beet field size statistics

highlighting that the average and third quartile fall below 4 hectares. Hence, most field patches contain substantial zero-padding, resulting in large regions of black pixels.

During processing, the SatMAE model further divides each (64, 64) patch into (16, 16) sub-patches. Due to the prevalence of black padding, the sub-patches might consist a majority of zero values. With a masking ratio of 75%, a large fraction of the sub-patches (including possibly black ones) are masked, and the model is trained to reconstruct them using only the unmasked subset. This might likely contribute to the black reconstructions, as the model might learn to reproduce black pixels present in both visible and masked regions.

Unlike the 3D_AE_B10 model, which discards empty sub-patches during preprocessing and uses only partially or fully filled $(4, 4)$ sub-patches for training, the SatMAE model uses the entire $(64, 64)$ field patches for modeling and reconstruction. As such, for SatMAE, we could not remove the zero-valued sub-patches without disrupting the structural consistency required by the architecture. This limitation might constrain the model's ability to focus on meaningful content.

Further experiments are needed to investigate this issue. For example, using a smaller sub-patch size (less than 16) may help preserve more informative spatial details. Additionally, the attention mechanism within the ViT blocks could be adapted to ignore the zero pixels. Other potential improvements include fine-tuning hyperparameters or modifying the reconstruction loss to address non-informative sub-patches more effectively.

Overall, the SatMAE model, as implemented in this study, was unable to learn meaningful representations of the input data, likely due to challenges introduced by zero-padding and field size variability. Nonetheless, the experiment offered a valuable learning experience, deepening our understanding of the complexities involved in applying masked autoencoders to satellite images. We believe that with further architectural adjustments and preprocessing strategies, the masked autoencoder paradigm holds promise for agricultural tasks using SITS data.

8 Conclusions

We conclude this thesis by summarizing the key tasks performed and the main insights gained throughout the study. This is followed by a discussion of limitations and potential future work, including enhancements to the current stress detection system and possible extensions building upon its foundation. Finally, we reflect on the personal takeaways from this journey.

8.1 Summary and key insights

Sugar-beets are a vital crop for sugar production, yet their cultivation has faced increasing challenges due to adverse environmental conditions and disease outbreaks. Traditionally, farmers rely on manual inspection to assess crop health. However, given the large size of sugar-beet fields—ranging from 0.14 to 52.44 hectares—this approach is labor-intensive and inefficient. Due to resource constraints, small-scale farmers often cannot afford high-resolution images or ground-truth labels, making supervised or semi-supervised learning approaches impractical. To address this, we adopted a fully unsupervised approach that leverages publicly available Sentinel-2 satellite images, which offers rich spectral information and frequent temporal coverage.

This thesis started by clearly defining the problem of stress detection in sugar-beet fields and outlining a strategy for addressing it using unsupervised learning. We iteratively reviewed relevant literature in the remote sensing for agriculture domain, drawing inspiration and insights that informed the design and continuous refinement of our stress detection system. Understanding the structure and challenges of Satellite Image Time Series (SITS) data was critical. We developed a preprocessing pipeline to extract sugar-beet fields from Sentinel-2 images and selected cloud-free observations that best represented key stages of the sugar-beet growth cycle. We then investigated a range of feature extraction techniques—from basic statistical features to autoencoders—aimed at capturing the spectral and temporal dynamics of the sugar-beet fields, which were divided into smaller sub-patches for localized stress detection. In particular, we focused on 3D convolutions and temporal encoding strategies, which are well-suited for time-series imagery. Due to limited availability of labeled data, clustering was used as a downstream task to categorize stressed and healthy fields. To bridge the gap between

sub-patch-level predictions and field-level ground truth, we implemented a thresholding mechanism.

Additional experiments explored the impact of input configurations, including reducing the number of spectral bands, using vegetation indices, and varying the sub-patch size and sub-patch-to-patch threshold. Finally, we tested the generalizability of our system on the more challenging 2024 sugar-beet season, which featured adverse weather, severe stress events, and limited usable temporal instances.

The key findings of the thesis are as follows:

- **Complexity of satellite data:** Satellite data is inherently complex and requires extensive preprocessing. Careful selection of spectral bands and temporal instances that accurately represent the sugar-beet growth cycle is essential to ensure high-quality inputs for modeling.

- **Importance of temporal encodings:** This study highlights the effectiveness of combining temporal encodings with 3D convolutions. This approach captures the progression of crop health over time, enabling the model to differentiate between the growth patterns of stressed and healthy fields more accurately.

- **Richness of raw spectral data:** Using the raw spectral bands outperformed pre-computed vegetation indices, likely due to the retention of richer and nuanced vegetation signals, which can be lost when using derived indices.

- **Importance of using smaller sub-patches:** Using a small sub-patch size proved effective for handling the small, variable-sized fields. It allowed us to discard irrelevant sub-patches containing filler or black pixels, thereby improving the data quality for modeling.

- **Configurable sub-patch-to-patch threshold:** Having a configurable threshold provides control over the amount of stress required to label a field as stressed. Such flexibility is valuable during extreme conditions, like heavy rain or drought, which can impact the stress levels across the field.

- **Generalizable stress detection system:** We developed a fully functional stress detection system that can be executed via a Python script after simple path configuration. The system was validated on data from 2024 season, demonstrating generalization across years and environmental conditions.

- **Localized stress maps:** As part of the deliverable, we generate localized stress maps highlighting affected sub-patches within each sugar-beet field. After refining the stress detection system through downstream supervised tasks, these maps can

be directly utilized by farmers, offering actionable insights.

In summary, we developed an unsupervised and generalizable stress detection system for sugar-beet fields by leveraging the temporal and spectral richness of Sentinel-2 satellite data. This thesis emphasizes the importance of temporal encodings and lays a strong foundation for future work in stress detection for sugar-beet fields. It also contributes to advancing unsupervised representation learning techniques tailored to agricultural applications, paving the way for future innovations in agriculture.

8.2 Limitations

It is important to interpret the results presented in this thesis with caution. The evaluation relies on a limited set of ground truth labels, which may include inaccuracies—some fields labeled as healthy may actually show undetected signs of stress. As a result, the reported performance metrics might not fully reflect the true capability of the model. While the 3D_AE_B10 model with temporal encodings shows promise, drawing strong conclusions would require further validation using a larger and more reliable labeled dataset that supports downstream supervised tasks such as classification. Given the current limitations in labeled data availability, the results of this study should be considered exploratory. Nevertheless, they demonstrate the potential of unsupervised methods to generate actionable insights in scenarios with limited labeled data.

8.3 Future work

In this section, we outline possible directions for future work to enhance the stress detection system, as well as potential components that can be built upon the developed framework. These include improving model accuracy through supervised learning, enabling early stress detection, adapting the system for other crops, and exploring additional methods to handle variability in field sizes.

8.3.1 Supervised downstream tasks

Currently, due to limited availability of labels, we perform downstream clustering instead of a supervised method. Moreover, the ground truth labels are provided by farmers, who may miss stressed areas because of the large size of the sugar-beet fields. As a

result, fields labeled as healthy may still contain stressed sub-regions, making the labels somewhat unreliable.

Once sufficient and reliable labels are available, the stress detection system can be refined to use a supervised learning task such as classification instead of clustering. While clustering offers a valuable unsupervised approach in the absence of labels, it may lead to groupings that are difficult to interpret in our settings—particularly due to complex nature of satellite data. In contrast, classification enables the model to learn explicit mappings between input data and stress labels. This can lead to more accurate, interpretable, and actionable predictions, particularly beneficial for real-world decision-making in agricultural contexts.

8.3.2 Early stress detection in sugar-beets

A key direction for future work is enabling early detection of stress in sugar-beet fields. Currently, the system models temporal data spanning the full growth cycle—from June to September—using seven temporal instances. However, by the time stress is detected in September, the crops may already have suffered significant yield loss.

To facilitate timely intervention, the system could be adapted to detect stress earlier in the season. This can be achieved by reducing the temporal window to include only the early-to-mid growth stages, such as June, July, and mid-August. Different techniques for early detection can also be explored, such as next-timestep prediction, where the model learns to predict future spectral responses based on earlier observations. Significant deviations between the predicted and actual future values may indicate abnormal crop development, potentially signaling stress.

Modeling early-season data would enable identifying stress signals before irreversible damage occurs, allowing farmers to respond proactively with targeted measures. Such an approach could significantly improve the practical utility of the system by shifting its focus from post-hoc analysis to early warning and prevention.

8.3.3 Applying the stress detection system to other crops

The current stress detection system has been developed specifically for sugar-beet fields, functioning in a fully unsupervised manner using Sentinel-2 satellite images. An interesting future direction would be to apply this system to other crops. For successful generalization, the target crop should ideally share a similar growth cycle and exhibit stress patterns comparable to sugar-beets—winter wheat being a potential candidate.

While the sowing-to-harvest timeline may differ, the system can be adapted by modifying the number of temporal instances and adjusting the crop growth period within the configuration file. This flexibility opens the possibility of building a stress detection framework that is not only generalizable across years but also across crops with similar phenological behavior.

8.3.4 Temporal interpolation to address cloud cover

In this thesis, we select a fixed number of temporal instances for all sugar-beet fields, ensuring they are completely cloud-free. However, this approach limits the use of the rich temporal data available from Sentinel-2. Additionally, sugar-beet fields without the required number of cloud-free observations are excluded from the dataset, reducing spatial coverage.

To address this, future work can explore temporal interpolation techniques such as linear interpolation [60] or Savitzky-Golay filtering [62]. Linear interpolation estimates missing values by assuming a straight-line change between adjacent cloud-free observations. Savitzky-Golay filters, on the other hand, perform local polynomial regression to smooth and fill gaps in time series data while preserving important features like peaks and trends. By incorporating such interpolation techniques during preprocessing, sugar-beet fields with partially clouded temporal sequences can still be used. This would enable the construction of bigger temporal stacks and increase the temporal resolution of model inputs, improving the ability to capture nuanced growth patterns and stress signals over time.

8.3.5 Addressing the field-size variability

As discussed earlier, sugar-beet fields vary widely in size—some extending up to 50 hectares. To handle this, we currently zero-pad all fields to a uniform input size, which introduces non-informative pixels and may introduce bias during modeling. To mitigate this, we adopt a sub-patch-based modeling approach, where smaller sub-patches are extracted. Sub-patches that are only partially filled with field pixels are completed using the average of the valid pixels within them.

While this strategy allows us to model variable-sized fields and also produce sub-patch-level predictions, it comes with both strengths and limitations. A major strength is the ability to generate localized stress maps, helping farmers identify specific areas of concern within large fields, thereby improving the actionability of the system. However, using averaged values in partially empty sub-patches can dilute the spatial precision of the model. Moreover, obtaining fine-grained ground truth labels is often impractical

for small-scale industries and farmers, thus limiting evaluation to coarser, field-level labels.

Future work could explore more advanced techniques for handling irregular field geometries, such as dynamic attention mechanisms that adapt to field shape and content, inspired by [66]. Such approaches may improve the robustness and reduce reliance on padding, further enhancing the generalizability of stress detection across diverse agricultural landscapes.

8.3.6 Training data availablity

Currently, we work with a limited amount of unlabeled data—1228 sugar-beet fields from the 2019 season and 1491 from the 2024 season. In future scenarios with access to larger and more diverse datasets, more advanced feature extraction methods could be employed. Such methods, particularly those based on attention mechanisms, have the potential to capture complex spatial and temporal dependencies in multispectral satellite imagery, improving the quality of learned representations. The SatMAE architecture, which incorporates ViT blocks, can be re-explored in the future by addressing the issues identified in this thesis (black reconstructions), potentially leading to improved performance with richer data availability.

Pursuing these future directions will significantly advance the development of reliable, scalable, and adaptable stress detection frameworks, enabling earlier and more effective decision-making in agricultural tasks.

8.4 Personal takeaways

Through this thesis, I was introduced to the fascinating domain of multispectral satellite imagery—a field that combines technical depth with meaningful real-world impact. Working with satellite data allowed me to appreciate its richness and complexity, as well as the unique challenges it poses. This experience ignited a lasting curiosity in remote sensing and motivated me to explore how machine learning and deep learning can unlock the full potential of SITS datasets.

One of the most valuable lessons I learned was the importance of clearly defining the problem before diving into implementation. Creating a structured plan and timeline helped me stay focused and make consistent progress. Developing the stress detection system was an iterative and exploratory process. Regular feedback and continual refinement helped me adopt a more agile and adaptive mindset.

The most profound insight was recognizing how real-world projects differ from standard academic datasets—real data is often messy, incomplete, and requires thoughtful handling. Navigating these challenges not only strengthened my problem-solving skills, but also gave me greater confidence in working with satellite data.

Ultimately, this thesis inspired me to continue exploring applications of satellite data in my future career, and I am excited to further contribute to this impactful field.

Bibliography

[1] *3D Convolutions*. URL: https://pytorch.org/docs/stable/generated/torch.nn.Conv3d.html.

[2] *8-Connectivity between Pixels*. URL: https://en.wikipedia.org/wiki/Pixel_connectivity#8-connected.

[3] Osama Abu Abbas. "Comparisons between data clustering algorithms." In: *International Arab Journal of Information Technology (IAJIT)* 5.3 (2008).

[4] *Agriculture Revolutionized via Satellite Imaging*. URL: https://farmonaut.com/remote-sensing/revolutionizing-agriculture-how-ai-powered-satellite-monitoring-is-transforming-farming-2/.

[5] *AI Stack Exchange: 1D Convolutions*. URL: https://ai.stackexchange.com/questions/28767/what-does-channel-mean-in-the-case-of-an-1d-convolution.

[6] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. "Vivit: A video vision transformer". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 6836–6846.

[7] *Bayerisches Landwirtschaftliches Wochenblatt: Gummirüben*. URL: https://www.wochenblatt-dlv.de/feld-stall/pflanzenbau/gummirueben-diese-neue-krankheit-befaellt-gerade-ganze-ruebenfelder-574405.

[8] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[9] European Commission. *Crop Productions and Plant pased Products Sugar*. URL: https://agriculture.ec.europa.eu/farming/crop-productions-and-plant-based-products/sugar_en.

[10] Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. "Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 197–211.

[11] Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. "Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 197–211.

[12] *Connected Component Labeling*. URL: https://neubias.github.io/training-resources/connected_components/index.html.

[13] Sentinel Copernicus. *Copernicus Sentinel-2 Collection 1 MSI Level-2A (L2A)*. URL: https://sentinels.copernicus.eu/sentinel-data-access/sentinel-products/sentinel-2-data-products/collection-1-level-2a.

[14] *Copernicus Sentinel-2*. URL: https://dataspace.copernicus.eu/explore-data/data-collections/sentinel-data/sentinel-2.

[15] *Crop Science Bayer: Beet Rust*. URL: https://www.cropscience.bayer.co.nz/pests/diseases/beet-rust.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[17] *Enhanced Vegetation Index Resistance*. URL: https://www.earthdata.nasa.gov/topics/biosphere/enhanced-vegetation-index-evi.

[18] *Farming UK: Cercospora in sugar beet*. URL: https://www.farminguk.com/news/first-cases-of-fungicide-resistant-cercospora-found-in-uk-sugar-beet_46086.html.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[20] Kerstin Gröll, Simone Graeff, and Wilhelm Claupein. "Use of Vegetation indices to detect plant diseases". In: *Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten–Referate der 27. GIL Jahrestagung*. Gesellschaft für Informatik e. V. 2007, pp. 91–94.

[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked autoencoders are scalable vision learners". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009.

[22] Sentinel Hub. *Sentinel-2 Collection Scene Classification Map*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/scene-classification/.

[23] *ISRO RISAT 1 Satellite*. URL: https://www.isro.gov.in/RISAT_1.html.

[24] Barry J. Jacobsen. *What's New in Sugarbeet Disease Management*. 2013. URL: https://www.sugarproducer.com/2013/04/whats-new-in-sugarbeet-disease.

[25] Shunping Ji, Chi Zhang, Anjian Xu, Yun Shi, and Yulin Duan. "3D convolutional neural networks for crop classification with multi-temporal remote sensing images". In: *Remote Sensing* 10.1 (2018), p. 75.

[26] Sachin D Khirade and Amit B Patil. "Plant disease detection using image processing". In: *2015 International conference on computing communication control and automation*. IEEE. 2015, pp. 768–771.

[27] *Label Studio Software*. URL: https://labelstud.io/.

[28] Zhouhan Lin, Yushi Chen, Xing Zhao, and Gang Wang. "Spectral-spatial classification of hyperspectral image using autoencoders". In: *2013 9th international conference on information, Communications & Signal Processing*. IEEE. 2013, pp. 1–5.

[29] A-K Mahlein, U Steiner, H-W Dehne, and E-C Oerke. "Spectral signatures of sugar beet leaves for the detection and differentiation of diseases". In: *Precision agriculture* 11 (2010), pp. 413–431.

[30] CL Meneses-Tovar. "NDVI as indicator of degradation". In: *Unasylva* 62.238 (2011), pp. 39–46.

[31] Ihab S Mohamed. "Detection and tracking of pallets using a laser rangefinder and machine learning techniques". PhD thesis. European Master on Advanced Robotics+(EMARO+), University of Genova, Italy, 2017.

[32] *Moisture Stress Index*. URL: https://marketplace-portal.dataspace.copernicus.eu/catalogue/app-details/2.

[33] *MSI range*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/msi/.

[34] *NASA TERRA Satellite*. URL: https://terra.nasa.gov/.

[35] *PlanetScope Data*. URL: https://earth.esa.int/eogateway/missions/planetscope.

[36] Draško Radovanović and Slobodan Đukanovic. "Image-based plant disease detection: a comparison of deep learning and classical machine learning algorithms". In: *2020 24th International conference on information technology (IT)*. IEEE. 2020, pp. 1–4.

[37] Muhammad M Raza, Chris Harding, Matt Liebman, and Leonor F Leandro. "Exploring the potential of high-resolution satellite imagery for the detection of soybean sudden death syndrome". In: *Remote Sensing* 12.7 (2020), p. 1213.

[38] *Representation Learning*. URL: https://paperswithcode.com/task/representation-learning.

[39] *Representation Learning for Engineers*. URL: https://blog.fastforwardlabs.com/2020/11/15/representation-learning-101-for-software-engineers.html.

[40] *RGB Imag Composition*. URL: https://anderfernandez.com/en/blog/how-to-create-convolutional-neural-network-keras/.

[41] T Rumpf, A-K Mahlein, U Steiner, E-C Oerke, H-W Dehne, and Lutz Plümer. "Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance". In: *Computers and electronics in agriculture* 74.1 (2010), pp. 91–99.

[42] *Scipy Binary Erosion*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.binary_erosion.html.

[43] *Sentinel-2 band names, wavelengths, and resolutions*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/.

[44] *Sentinel-2 Image*. URL: https://learn.opengeoedu.de/en/fernerkundung/vorlesung/copernicus/Sentinel-2-Teil-1.

[45] *Sentinel-2 Processing Levels*. URL: https://sentiwiki.copernicus.eu/web/s2-processing.

[46] Leninisha Shanmugam, AL Agasta Adline, N Aishwarya, and Gokulnath Krithika. "Disease detection in crops using remote sensing images". In: *2017 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*. IEEE. 2017, pp. 112–115.

[47] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[48] *Skimage function for getting Region Properties*. URL: https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.regionprops.

[49] *Skimage function for labeling Connected Components*. URL: https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.label.

[50] Suedzucker. *The Sugar Beet: A Sustainable All-Rounder Plant*. URL: https://www.suedzucker.com/sustainability/sugar-beet/.

[51] *Sugar Beet Cultivation*. URL: https://www.zuckerverbaende.de/anbau-und-verarbeitung/ruebenanbau/zuckerruebenanbau/.

[52] *Sugar-beets: Landesbetrieb Landwirtschaft Hessen*. URL: https://llh.hessen.de/pflanze/marktfruchtbau/zuckerrueben/.

[53] Michail Tarasiou, Erik Chavez, and Stefanos Zafeiriou. "Vits for sits: Vision transformers for satellite image time series". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

[54] *Understanding Vegetation Indices*. URL: https://help.dronedeploy.com/hc/en-us/articles/1500004860841-Understanding-Vegetation-Indices.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[56] Brandon Victor, Aiden Nibali, and Zhen He. "A systematic review of the use of Deep Learning in Satellite Imagery for Agriculture". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2024).

[57] Yi Wang, Conrad M Albrecht, Nassim Ait Ali Braham, Lichao Mou, and Xiao Xiang Zhu. "Self-supervised learning in remote sensing: A review". In: *IEEE Geoscience and Remote Sensing Magazine* 10.4 (2022), pp. 213–247.

[58] *Wikipedia: Enhanced Vegetation Index*. URL: https://en.wikipedia.org/wiki/Enhanced_vegetation_index.

[59] *Wikipedia: Gaofen imaging satellites*. URL: https://en.wikipedia.org/wiki/Gaofen.

[60] *Wikipedia: Linear Interpolation*. URL: https://en.wikipedia.org/wiki/Linear_interpolation.

[61] *Wikipedia: Normalized Difference Vegetation Index*. URL: https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index.

[62] *Wikipedia: Savitzky Golay filters*. URL: https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter.

[63] *Wikipedia:Histogram Square Root rule*. URL: https://en.wikipedia.org/wiki/Histogram#Square-root_choice.

[64] *Wikipedia:Histograms*. URL: https://en.wikipedia.org/wiki/Histogram#.

[65] *Wikipedia:PCA*. URL: https://en.wikipedia.org/wiki/Principal_component_analysis.

[66] Fei Wu, Feng Chen, Xiao-Yuan Jing, Chang-Hui Hu, Qi Ge, and Yimu Ji. "Dynamic attention network for semantic segmentation". In: *Neurocomputing* 384 (2020), pp. 182–191.

[67] Cheng Xiang, Dalei Wang, Yue Pan, Airong Chen, Xiaoyi Zhou, and Yiquan Zhang. "Accelerated topology optimization design of 3D structures based on deep learning". In: *Structural and Multidisciplinary Optimization* 65.3 (2022), p. 99.

[68] Run Yu, Youqing Luo, Quan Zhou, Xudong Zhang, Dewei Wu, and Lili Ren. "Early detection of pine wilt disease using deep learning algorithms and UAV-based multispectral imagery". In: *Forest Ecology and Management* 497 (2021), p. 119493.

[69] Li Zhang and Andreas CW Baas. "Mapping functional vegetation abundance in a coastal dune environment using a combination of LSMA and MLC: a case study at Kenfig NNR, Wales". In: *International Journal of Remote Sensing* 33.16 (2012), pp. 5043–5071.

[70] Chunhui Zhao, Hao Cheng, and Shou Feng. "A spectral–spatial change detection method based on simplified 3-D convolutional autoencoder for multitemporal hyperspectral images". In: *IEEE Geoscience and Remote Sensing Letters* 19 (2021), pp. 1–5.

List of Figures

Appendix

Image related terminologies

Below, we present visualizations of Sentinel-2 data corresponding to the image-related terminology used throughout this thesis.

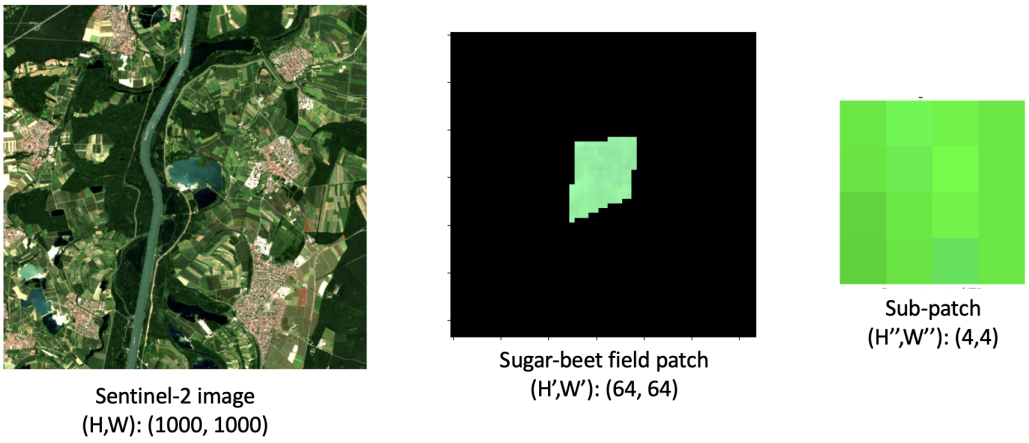


Figure 1: Sentinel-2 data according to the terminologies, shown for a single temporal instance using RGB composite.

Sentinel-2 bands

Visualizations of individual Sentinel-2 image bands (excluding Bands 1, 9, and 10) are shown for a single temporal instance of the entire Sentinel-2 image, and sugar-beet field patch.

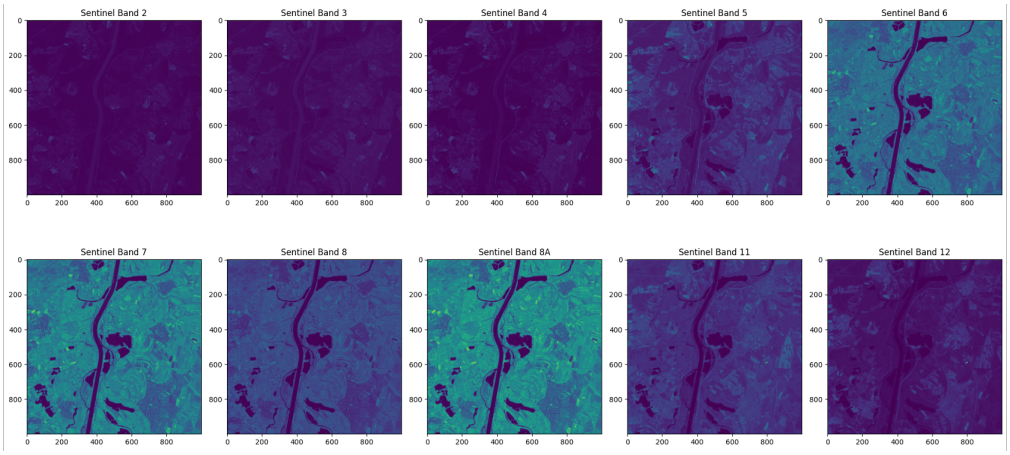


Figure 2: Sentinel-2 image bands (except bands 1,9 and 10).

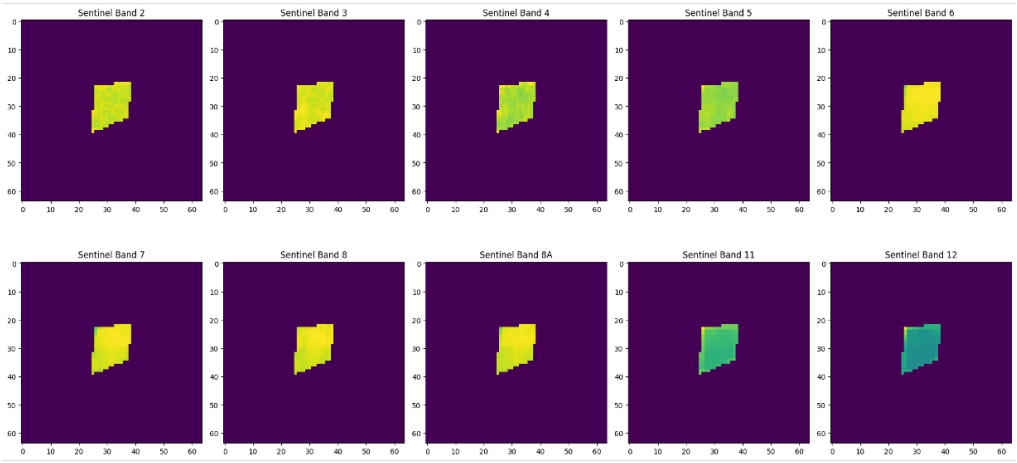


Figure 3: Sentinel-2 bands for sugar-beet field patch (except bands 1,9 and 10).

Reconstructions

Visualizations of sub-patch-level reconstructions, along with their mappings back to the patch level. Original and reconstructed images are shown below for the models 2D_AE_B10 and 3D_AE_B10, both with temporal encodings.

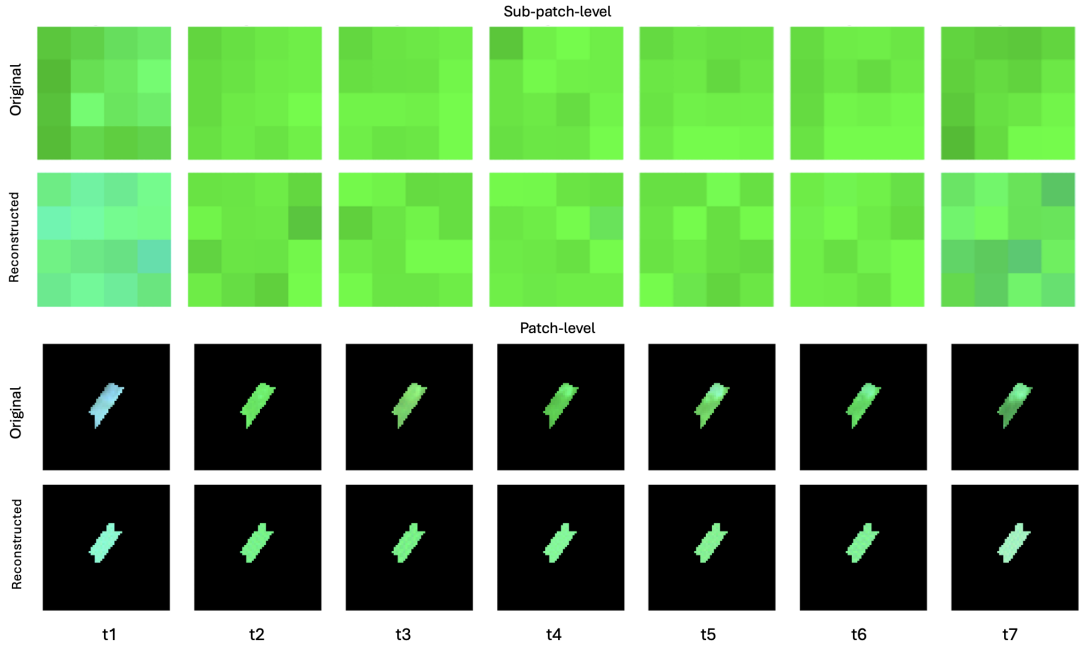


Figure 4: Original and reconstructed sub-patches along with their corresponding patch-level mappings for 2D_AE_B10 with temporal encodings.

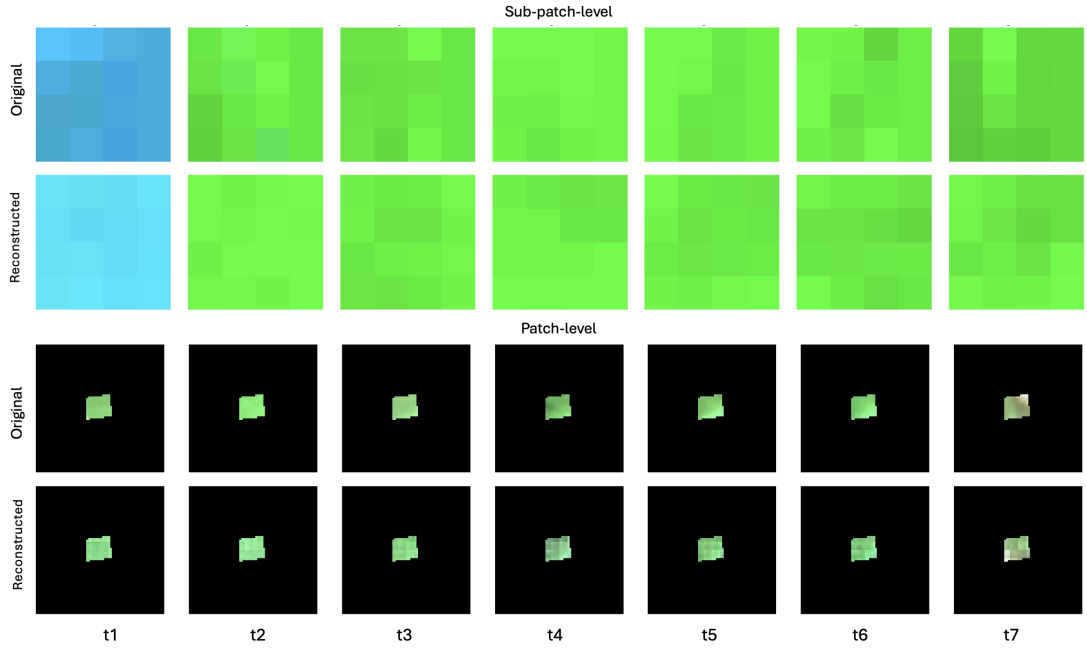
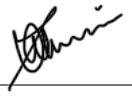


Figure 5: Original and reconstructed sub-patches along with their corresponding patch-level mappings for 3D_AE_B10 with temporal encodings.

Declaration on oath

I hereby certify that I have written my master thesis independently and have not yet submitted it for examination purposes elsewhere. All sources and aids used are listed, literal and meaningful quotations have been marked as such.

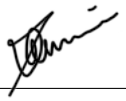


August 2, 2025, Bhumika Laxman Sadbhave

Consent to plagiarism check

I hereby agree that my submitted work may be sent to PlagAware (https://my.plagaware.com/) in digital form for the purpose of checking for plagiarism and that it may be temporarily (max. 5 years) stored in the database maintained by PlagScan as well as personal data which are part of this work may be stored there.

Consent is voluntary. Without this consent, the plagiarism check cannot be prevented by removing all personal data and protecting the copyright requirements. Consent to the storage and use of personal data may be revoked at any time by notifying the faculty.



August 2, 2025, Bhumika Laxman Sadbhave