Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Science and Technology - REALTEK

Multi-spectral UAV Data Analysis for Wheat Yield Prediction: A Deep Learning Approach

Muhammad Ashar
Master of Science in Data Science

Contents

List of Figures

List of Tables

Acknowledgements

I extend my heartfelt gratitude to my supervisor, Ingunn Burud and my co-supervisor, Sahameh Shafiee, for their unwavering guidance, support, and invaluable mentorship throughout this journey. Their expertise, encouragement, and constructive feedback have been instrumental in shaping this thesis and my growth as a student.

I would also like to express my deepest appreciation to my parents for their endless love, encouragement, and belief in my abilities. Their unwavering support and sacrifices have been the bedrock of my academic pursuits, and I am forever indebted to them.

Thank you for being my pillars of strength, guiding lights, and constant sources of inspiration. This achievement would not have been possible without your unwavering belief in me.

ABSTRACT

Through the use of deep learning models and a comprehensive dataset of multispectral time series data collected by unmanned aerial vehicles (UAVs), this thesis aims to estimate the grain yield of a wheat field. The major goal is to contribute to agricultural advancements by creating precise and effective predictive models that may help farmers and decision-makers to manage crop production and maintain food security. To find the best candidate for this prediction task, a variety of machine learning and deep learning models are investigated, including Gradient Boosting Regressor (GBR), Deep Neural Networks (DNNs), and Bidirectional Long Short-Term Memory (BiLSTM) networks. The dataset's completeness and quality must be guaranteed if the model is to succeed. After some visual inspection of the data and its time series aspects, different pre-processing methods such as resampling, imputation, interpolation and second order differencing are used to ensure a fair chance for the models to learn. The study demonstrates how modern technology has the potential to change agricultural practices by utilizing the capabilities of multispectral UAV data. The ability to anticipate grain yield accurately opens up possibilities for improving crop management, resource allocation, and environmentally friendly agricultural methods. Despite having difficulties with hyperparameter tuning, resolving concerns about overfitting, and dealing with missing values, the findings yield good results and sets the groundwork for a new era of agricultural forecasting where multispectral UAV data and deep learning intersect to enable informed decision-making and sustainable farming methods.

1. INTRODUCTION

1.1. Background

A major component of human nutrition, wheat is a staple cereal crop that provides the main source of food for millions of people worldwide. Wheat continues to play a significant role in modern agriculture, meeting the world's rising food need while also having historical significance that dates back to ancient civilizations. In our daily lives, the food items that are considered most nutritious and beneficial for the human will have the use of wheat in one way or another. From bakery items using wheat flour to products made of wheat starch like energy drinks, food additives and animal feed, it can be established that wheat is one of the most fundamental requirement of our diet. The global demand of agricultural crops is projected to almost double by the year 2050 (Tilman, et al., 2011) [1]. On the other hand, there is another study that states: Total global crop production has only increased by 28% in between the years 1985 to 2005. In about 24-39% of staple crops growing areas, grain yield either never improved, is decreasing, or has levelled out (D K Ray, et al., 2012) [2]. These studies emphasize on the need of innovative solutions to improve the production process of wheat grain. There are multiple traits of a wheat plant that can be studied to help with this purpose. It can include observing and studying about the plant's physical appearance and looking for obvious signs of any potential disease. It is crucial to accurately forecast wheat grain yield because it helps farmers and agricultural officials to manage crop production, make use of resources in the best possible manner, and ensure food security. However, the traditional approaches rely on manual observations and measurements, which are time-consuming and prone to inaccuracy. These difficulties can be overcome and precise estimates of the wheat yield can be achieved, thanks to the developments in machine learning algorithms and remote sensing technologies, such as the utilization of UAV-based multispectral data. The study of different traits or characteristics of the plant is also known as Plant Phenotyping.

1.2. Multispectral UAV Data and Deep Learning

In recent years, technological advancements have revolutionized agricultural practices, ushering in new tools to tackle age-old challenges. Utilizing unmanned aerial vehicles (UAVs) with multispectral sensors is one such invention. This state-of-the-art method enables the quick acquisition of high-resolution data, collecting many spectral bands outside the range of the human eye. UAVs provide a bird's-eye perspective of the crop's health and development by collecting multispectral time series data over the wheat

fields, offering a wealth of information for crop monitoring and yield prediction. Some branches of machine learning such as Deep Learning has revolutionized a number of industries by proving to be exceptionally adept at evaluating large, complicated datasets. Significant advances in image identification, natural language processing, and now agricultural applications have all been made possible by machine learning's ability to extract complex patterns and characteristics from enormous amounts of data.

1.3. Problem Statement

The main objective of this thesis is to use multispectral UAV data and deep learning techniques to create an accurate and dependable model for predicting wheat grain yield. Multiple traits including Grain Yield, Days to Maturity and Days to heading were observed calculated manually by the other researchers in the farm. The trait of interest for this study is Grain Yield that can be defined as the total yield of the field in a specific area. The units originally used while calculating were g/m^2 (grams per meter square) but then were converted to t/ha (tonnes per hectare) as per the usual standard in the industry. The dataset is vast and complex because there are multiple observations for each plot in the field. Cleaning data, further pre-processing and time series analysis can prove to a challenge. Since the output variable, in this case grain yield has continous and positive numerical values, it makes it a regression problem.

1.4. Scope and Limitations

The scope of this study is limited to the prediction of wheat grain yield using multispectral UAV data and deep learning models, alongwith Time Series Analysis. The time series data used in this study is a part of the vPheno (Virtual Phenomics) project funded by the Research Council of Norawy. It is made up of multispectral images that were recorded using a UAV-mounted multi-sprectral camera in 5 bands: Red, Green, Blue, Near Infrared (NIR), and RedEdge. These images were taken on a pre-defined automated path and stitched together. To cut down on noise and produce more accurate statistics, the median of these bands was taken. The study's limitations can be the possibility of mistakes in the manual measurements of grain yield. Additionally, changes in weather patterns, soil quality, and other environmental factors that can impact crop growth may as well have an impact on the accuracy of the deep learning models.

1.5. Workflow

A wide range of data pre-processing techniques have been used to potentially generate accurate predictions utilizing the dataset. This includes the removal of outliers, handling of missing values, resampling the data to a consistent frequency, transformation to 3D tensors to represent the data in a format suited for deep learning models, and standardization of the data to bring all features to a common scale are some of these. In this study, a number of deep learning and machine learning models have been utilized for predicting wheat grain yield. Gradient Boosting Regressor, LSTMs, Bi-LSTMs, and GRU are among the models tested. These models were selected because they can work with time-series data and can discover intricate patterns and relationships in the data. In addition to the machine learning models, time series analysis has also been taken into consideration. In order to analyze the time series data for stationarity, it had to be decomposed into trend, seasonality, and residuals. The effect of trend and seasonality was then reduced by differencing the time series data. These procedures were crucial in ensuring the accuracy and dependability of the prediction models.

1.6. Conclusion

The introduction of this thesis sets the stage for an exciting journey into the realm of wheat grain yield prediction, highlighting the importance of wheat and its significance in modern agriculture. The following chapters explore the theoretical foundations, data pre-processing, deep learning models, and analysis of outcomes, yielding insightful knowledge to support agricultural decision-making and contribute to a future that is more sustainable.

2. THEORY

2.1. Plant Phenotyping

Plant phenotyping is the study of the interaction of genetic characteristics of plants with the environment and their resulting effects on the development of more desirable traits (Minnervini, et al., 2013) [3]. It can include many morphological, physiological, biochemical, and molecular characteristics of a plant. The growth of plant phenotyping as a scientific field has been characterized by significant developments throughout its lengthy history. Early plant phenotyping pioneers helped enhance our knowledge of plant biology and laid the path for the creation of cutting-edge methods and technology.

Phenotyping is a crucial process for crop improvement because it enables us to spot desirable traits in plants that can help them to grow better in future. These traits include, but are not limited to, plant height, width, leaf area, leaf shape, chlorophyll content, disease resistance and yield, etc. It has numerous applications in crop development and agricultural research. The production of crop varieties with enhanced yield, stress tolerance, and disease resistance is made possible by its assistance in breeding programs, genetic mapping, and trait identification. Research on plant phenotyping encounters difficulties with data collection, analysis, and interpretation. Researchers have difficulties when integrating and standardizing multi-scale phenotypic data. Large dataset administration and analysis also call for sophisticated computational methods and tools. But due to interdisciplinary research efforts and technological breakthroughs, the subject of plant phenotyping continues to evolve. Future research is showing promise in areas including high-throughput phenotyping, bioinformatics, machine learning, and genomics. These developments have the potential to speed up agricultural improvement and support sustainable farming.

2.2. Multispectral Imaging and use of UAVs

Multi-spectral imaging is a type of imaging technology that captures images of an object or scene in multiple wavelengths or colors of light. In contrast to conventional color photography, which only captures images in the three primary colors of red, green, and blue, it consists of capturing images in a wide variety of bands, often from ultraviolet to infrared. Multi-spectral imaging relies on specialized sensors or cameras that can recognize and capture light at different wavelengths. Each band corresponds to a specific range of wavelengths. The three primary colors that have a wavelength ranging roughly from 400nm to 700nm (nanometres), can also be seen by a naked human eye. But the wavelengths captured by a multi-spectral camera, otherwise not visible to a human eye, can be very useful in various fields like medicine, environmental monitoring and crop assessment. Especially when studying wheat fields, there are some heat signature and chlorophyll content that can be easily captured and then studied using one of these cameras. By analyzing the specific reflectance or absorbance of an object, it is possible to identify and analyze this object (Humboldt, 2018)[4]. While it has many benefits, there are some drawbacks and limitations too. Multispectral data collecting and processing calls for specific tools, knowledge, and computational power. To provide precise and trustworthy results, careful calibration and correction of the obtained images are required. Moreover, the quality and interpretation of multispectral data may be impacted by atmospheric factors, sensor noise, and spectral resolution limitations.

Capturing images of wheat fields that are usually huge, can be an intensive work altogether. Luckily with the rise in popularity of Unmanned Aerial Vehicles (UAVs), it has become really feasible to collect such data with great efficiency. A UAV is a type of aircraft that doesn't have a human pilot onboard. It can be designed to operate autonomously or under remote direction to carry out a variety of activities. UAVs frequently come with a variety of sensors, cameras, and tools attached, that enable them to collect data in the air. These UAVs can be equipped with multi-spectral cameras and a path can be pre-programmed to them so they can automatically capture multiple images of the wheat plots across the whole field and stitch them together for further processing.

2.3. Time Series Data

The data is usually stored in the form of a matrix (or rows and columns). This can be any excel file or a csv (comma-separated values) file. It is also known as a dataset. The columns are more commonly called features while rows are called samples. If the order of the data matters, then it is called a Sequential Data. Time series is a special type of sequential data where the axis along the data is referred to as time (Raschka, et al., 2019) [5]. In this type of data, the values of a variable of interest are recorded over a specific time period, such as hours, days, weeks, months, or years. For example if the wheat images taken using a UAV equipped with a multi-spectral camera at an interval of every 5 days, the organized and sequential dataset, along with the date of each observation can now be called a time series data. If only one variable is observed throughout the time period, it is called 'Univariate Time Series'. On the contrary, if multiple variables are observed throughout a specific time peroid, it is called 'Multivariate Time Series'. The time series data has certain properties such as trend and seasonality which means that the data follows similar pattern over a specific period of time. These so called patterns can then be used to analyze or predict certain features of our interest. In a wheat field, analyzing the crops over a specific time period frequency can yield with better results as compared to data collected at a single time point.

2.4. Time Series Analysis

Time series analysis is the process of analyzing and modeling time series data to understand its underlying structure or to make predictions. It offers important insights into the behavior and features of data by evaluating temporal patterns and trends. The following terminologies of time series analysis are relevant for this study.

2.4.1. Decomposition

Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category. (Hyndman, et al. 2018) [6] A time series can be divided into its trend, seasonality, and residual (or error) components using a process called time series decomposition. The trend component captures the underlying rise or drop over time and shows the data's long-term direction or behavior. The repeating patterns that happen at regular intervals and are frequently correlated with calendar or seasonal events are captured by the seasonality component. Analysts can better comprehend the inherent temporal patterns and produce more precise forecasts or projections by isolating these components. The performance of machine learning models can also be improved by removing such components from the data. Residuals comprise the unexplained part of data variability, the part not accounted for by the trend and seasonal components (C. Deb, et al, 2017) [7].

2.4.2. Trend Analysis

A vital component of time series analysis is trend analysis, which focuses on detecting and deciphering the long-term patterns and behaviors displayed by a dataset over time. Understanding the overall trend and trajectory of the data, whether it is rising, falling, or essentially stable. Finding whether the trends are linear, exponential, or involve other types of growth or decay, is the main purpose of trend analysis. Analysts can generate accurate predictions, evaluate the effects of particular variables, and spot potential turning points by spotting and modeling trends. The trend can be identified by taking the moving average of the data over a given period of time.

2.4.3. Seasonality Analysis

The process of identifying and simulating the periodic patterns in time series data is known as seasonality analysis. Seasonality can occur at different time scales, such as daily, weekly, or yearly cycles. Like the trend analysis, seasonal analysis can also prove to be helpful in making accurate predictions.

2.4.4. Stationarity

The concept of stationarity, which describes the property of time series data where the statistical aspects of the data remain constant across time, is another important concept in time series analysis. The mean and variance of a stationary time series are fixed. On

the other hand, non-stationary time series exhibit statistical characteristics that change over time, such as a fluctuating mean or variance. If a time series exhibits trend or seasonality, it is considered non-stationary. In both cases it is possible to transform the series to stationary by differencing it (Hyndman, 2018)[6].

2.4.5. Differencing

If a time series data is non-stationary, differencing is a typical technique to make it stationarity. In this technique, each observation is subtracted from the prior observation to produce the differenced time series. By calculating these differences, the residual component is taken into account, so the data noise and short-term variations are considered, while the influence of trends and seasonality is excluded due to a stabilized mean. This is called first-order differencing. This might not always completely remove the non-stationary components. By repeating the differencing operation in these circumstances, higher-order differencing can be carried out. The series that has already been differentiated, is differentiated again in second-order differencing, and so on. Higher-order differencing should be utilized with caution though, as it can result in the loss of important data and the introduction of unwanted noise.

2.5. Data Pre-processing

Data pre-processing is a crucial step before handling the data to any machine learning algorithm. It allows the data to be transformed into a format that can be easily understood and processed by the model. In this section, a number of data pre-processing methods relevant to this study are discussed including resampling, interpolation, outlier handling and other commonly used methods.

2.5.1. Data Cleansing and Outlier Detection

This involves dealing with missing values, eliminating outliers, and resolving flaws or inconsistencies in the data. Inconsistencies can be fixed by data validation and cleaning procedures, missing values can be imputed using the proper methods, outliers can be identified and either deleted or addressed. Outliers are data points that are significantly different from the rest of the data. Outliers must be handled carefully during pre-processing since they can significantly affect how well a machine learning model performs. Outliers are commonly handled by removing them from the dataset. However, if the outlier covers a significant portion of the data, this could be troublesome. Capping

the outlier numbers to keep them inside a predetermined range is another approach. One method to achieve this can be IQR Percentile.

2.5.2. Correlation

The statistical relationship between several features or variables in a dataset is referred to as correlation. It is employed to comprehend how modifications to one feature affect changes to another feature. Discovering patterns, dependencies, and correlations among the data features is mostly dependent on correlation analysis. This is an important step to make decision about model training and feature selection. If the correlation coefficient is close to 1, it indicates a strong positive correlation between the features. This means that as one feature increases, the other tends to increase as well. If the correlation coefficient is close to -1, it indicates a strong negative correlation between the features. This means that as one feature increases, the other tends to decrease. A correlation coefficient close to 0 suggests no significant linear relationship between the features. In this case, changes in one feature do not have a strong impact on changes in the other.

2.5.3. IQR Percentile

The Interquartile Range (IQR) Percentile technique is used to locate and deal with outliers in a dataset. A quartile is a form of quantile used in statistics that divides a dataset into four equal sections. A number below which 25% of the data falls is referred to as the first quartile (Q1), a value below which 50% of the data falls is referred to as the second quartile (Q2), and a value below which 75% of the data falls is referred to as the third quartile (Q3).
The range between a dataset's first and third quartiles is known as the Interquartile Range (IQR). A common technique for reducing outliers from a huge dataset is the IQR percentile. The lower and upper thresholds for the IQR percentile method are determined as Q1 - (1.5 IQR) and Q3 + (1.5 IQR), respectively. Outliers are any data points that are outside of this range and can be eliminated from the dataset. In contrast to approaches based on mean and standard deviation, which are susceptible to extreme values, the IQR Percentile method is robust against extreme values and offers a more accurate measurement for identifying outliers.

2.5.4. Resampling

Data Resampling is a technique used to change a time series dataset's frequency or structure. Changes to the time intervals or data aggregation to a new temporal resolution are necessary. Resampling is frequently used to account for irregular or irregularly spaced data points, align data to a specified time frame, or aggregate data from a higher frequency to a lower frequency (downsampling) or vice versa (upsampling). The specific criteria and data properties determine the resampling approach to use.
Downsampling, or decreasing the frequency of data points, involves aggregating or averaging multiple observations within a specific time interval to obtain a consolidated data point. When working with high-frequency data and trying to simplify computations or get a bigger picture of the overall trends in the data, this is helpful. However, because there are fewer data points, downsampling could lead to information loss.
Upsampling, or increasing the frequency of data points, involves interpolating or adding new data points to fill in the gaps between existing observations. To estimate the values between existing data points during upsampling, a variety of interpolation techniques, including can be used.

2.5.5. Imputation & Interpolation

Imputation is a technique used to fill in missing values using known data from the dataset. It utilizes statistical methods or models based on the available data to estimate the missing values. Imputation techniques utilise this knowledge to produce reasonable values for the missing data by taking into account the relationships between the variables. Imputation's objective is to provide a comprehensive dataset while maintaining the original data's statistical characteristics and patterns.
Interpolation is a technique that uses the presumption of a smooth or continuous relationship to estimate values between known data points. When dealing with ordered or sequential data, it is especially helpful. By taking into account the patterns and correlations found in the nearby data points, interpolation algorithms compute and fill in the missing values. The objective is to produce a comprehensive dataset while preserving the general structure and trends of the original data.
Both approaches seek to deliver a complete dataset without any null values, for additional analysis or modeling while maintaining the data's integrity and dependability. The type of data, the existence of patterns or linkages, and the specific goals of the analytic or modeling activity, all influence the decision between interpolation and imputation.

2.5.6. Reshaping to 3D Tensors

Some deep learning models require the data to be in a specific format. For instance, some Recurrent Neural Networks require the input data to be in the form of a 3D tensor so the model can capture the temporal dependencies and patterns present in the data. Before supplying the data to a recurrent neural network (RNN) or other models that require 3D tensor inputs, reshaping is employed to transform the data into the proper shape. The resulting 3D tensor structure represents a sequence of observations with multiple features at each time step. Each element in a 3D tensor is indexed by three dimensions: time steps, features, and samples. The data is arranged into a three-dimensional array in a 3D tensor.

2.5.7. Standardization and Normalization

Standardization and Normalization are rather important data pre-processing techniques that enhances the effectiveness of machine learning models. While normalization involves scaling the data such that it falls within a specific range, usually between 0 and 1, standardization scales the data so that it has zero mean and unit variance. When the scales or units of measurement of the features in the data differ, standardization is often used to prevent these features from having a disproportionately large impact on the model. We can make sure that each feature is given equal weight in the model by standardizing the data. On the contrary, when the data's range of values varies greatly, normalization is frequently utilized. We can make sure that all features have a comparable scale and can be directly compared by normalizing the data.

2.5.8. Train Test Split

In machine learning, a model is trained using a set of data to make predictions on new and unseen data. However, we must test our model on a different set of data in order to judge its effectiveness and make sure that it can generalize effectively to new data. This is where the concept of train-test split comes into play. The data is divided into two sets: a training set and a testing set. The model is trained on the training set, and its performance is assessed on the testing set. By evaluating the model's performance on the testing set, we can get an estimate of its generalization ability. If the model does well on the testing set, it will probably do well on fresh, untainted data. If the model doesn't perform well on the testing set, it may have overfitted to the training set, which means it has memorized the training set rather than learning from it.

2.6. Machine Learning

Machine Learning (ML) has a long history and has gone through various stages of development before becoming one of the most revolutionary areas in computer science. The origins of machine learning may be traced back to the 1950s, when pioneers like Alan Turing and Arthur Samuel investigated the idea of "learning machines." However, with the emergence of symbolic AI and rule-based systems in the 1980s, there was a tremendous advancement. As a subfield of artificial intelligence, machine learning focuses on developing algorithms and models that learn from data, making decisions and predictions without explicit programming (M Bishop, 2006)[8]. In many industries such as healthcare, finance, engineering and farming, machine learning has quickly proven itself to be an essential tool for resolving complicated problems. These algorithms can find patterns and relationships in the data that otherwise are not that obvious to humans, and use them to make predictions, classifications or some other decisions they were intended for. Machine Learning in turn, has other subcategories and terminologies that include regression, classification, clustering and deep learning. Each has its own set of algorithms and methodologies and are used for their own unique purposes.

Supervised machine learning is a fundamental paradigm where the model is trained on a labeled dataset, which means that both the input features and the matching output labels are given during training. The objective is for the model to discover the underlying correlations and patterns in the data, enabling it to make precise predictions on brand-new, untainted data. Tasks like classification, where the model divides data into predetermined classifications, and regression, where the model forecasts continuous numerical values, are examples of supervised learning. Supervised learning is ideally suited for a wide range of real-world applications, from image recognition and natural language processing to medical diagnosis and fraud detection, thanks to the availability of labeled data. Unsupervised machine learning, on the other hand, operates on an unlabeled dataset, where no equivalent output labels are present and just input features are accessible. Unsupervised learning's main goal is to find structures and patterns in the data without explicit instruction. In order to expose underlying links, the model aims to group together comparable data points or minimize the dimensionality of the data. Techniques for unsupervised learning are used for problems including feature learning, anomaly detection, and clustering. Since they can extract patterns and insights from unstructured, unlabeled data, they are especially helpful in situations where labeled data is hard to come by or expensive to acquire.

Two typical supervised learning problems in machine learning are classification and regression. In both cases, predictions regarding an output variable are made by taking into

consideration input variables, but the nature of the output variable differs. If the output variable is categorical or discrete, it makes it a classification problem. The main objective of a classification algorithm is to predict the class or category using unforeseen input data. Examples of classification can be, determining if an email is spam, categorizing good and bad reviews of a movie, etc. If the output variable is continuous or numerical, it makes it a regression problem. It also means that it can have any true value within a specified or unspecified range. A regression algorithm's objective is to forecast the value of the output variable using unforeseen input data. Examples can include estimating a house value based on its attributes, making stock market predictions and last but not the least, predicting grain yield of a wheat field, which is why this study can be recognized as a problem for supervised learning, more specifically Regression. There are many different type of regression models like Lasso Regression, Random Forest but for this study only Gradient Boosting Regressor (GBR) has been used to set-up a benchmark, since the main focus of this study was to develop a successful model using Deep Learning.

2.6.1. Gradient Boosting Regressor

Jerome Friedman first developed the Gradient Boosting Regressor (GBR) ensemble learning method in 2001. It is a type of ensemble machine learning algorithm used for regression problems. The main idea behind the workflow of GBR is to use simple models (like a decision tree), learn from its subsequent model, hence creating an overall strong learner by combining these so called weak learners. By learning form the errors of the prior models, GBR sequentially adds models to the ensemble after fitting the model to the training data. It accomplishes this by averaging the outputs of each model in the ensemble. Weights are chosen to reduce the difference between expected and actual values. GBR has proven to give the most accurate results amongst other different Machine Learning based regression models in this study which uses a similar variation of this data (Ijaz, 2019) [9]. This is the reason why GBR was chosen as a benchmark model for this study.

2.7. Deep Learning

The concept of an artificial neural network (ANN) was first invented in the 1940s, in an attempt to emulate the human brain's ability to solve complex problems. Over the years it has developed as a mathematical framework straying from its neurobiological inspiration, such that it is not correct to claim that the ANN learning mechanism mimics the one performed by human brain (F. Chollet, 2017)[10]. A neural network with several

layers, each of which is made up of a group of nodes or neurons that conducts a particular calculation on the input data, is the fundamental building block of deep learning. Such model trains itself by adjusting the weights and biases of the neurons in such a way that it minimizes the difference between the predicted output and the actual output. Utilizing multi-spectral time series data, deep learning has been used to estimate wheat trait predictions on many different studies, which has enabled farmers and researchers to make precise and timely decisions regarding these crops.

2.7.1. Deep Neural Networks

Deep Neural Networks (DNNs) are a powerful class of deep learning algorithms that have been used to tackle a broad range of challenging problems, including speech and picture identification, natural language processing, and even playing challenging games like Chess. These neural networks are made up of interconnected layers of artificial neurons that can be trained to recognize patterns and relationships in data. They are inspired by the structure and operation of the human brain. The diagram in Figure 1 shows an example of the work flow of a Deep Neural Network (Micheal Nielsen, 2015) [11]. The circles represent the nodes of the network. The input nodes are usually equivalent to the number of features being used as inputs. The hidden layers can be different for every model. Some models can have a lot of hidden layers and nodes depending on the complexity of the problem. The output layer similarly, is dependent on the number of output variables. In between these layers, the data is processed through so called Activation Functions that learn from previous iterations and try to improve their performance in the future iterations.

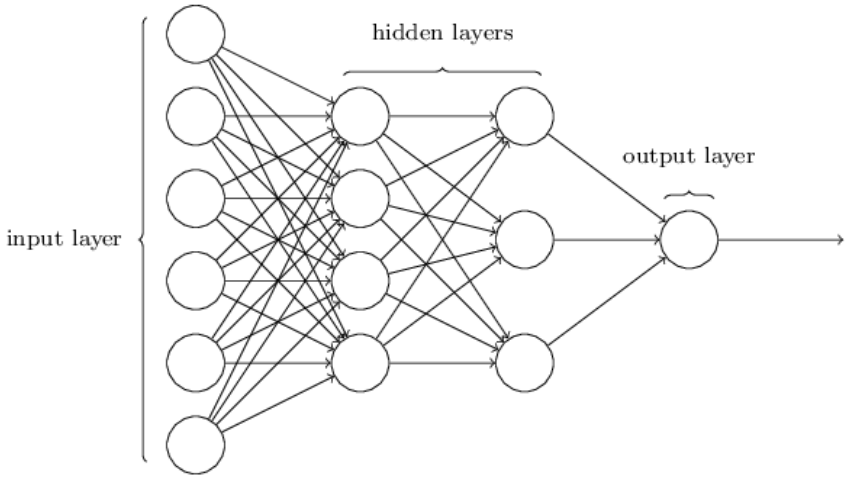


Fig. 1: General Diagram of a Deep Neural Network

Note: This image and book is allowed to be copied and shared as per the Creative Commons Attribution-NonCommercial 3.0 Unported License.

2.7.2. Recurrent Neural Networks

The history of Recurrent Neural Networks (RNNs) dates back to the 1980s when they were first proposed as a neural network architecture to handle sequential data. RNNs are advantageous for tasks involving sequential or time-series data because they feature loops that enable information to remain and be exchanged between time steps, in contrast to standard feedforward neural networks. RNNs can accept input sequences of varying length and capture dependencies between sequence members.
RNNs make use of the sequential character of time-series data to estimate wheat yield. The multispectral observations of the wheat field can be viewed as time steps, and the RNN processes these observations in order to detect temporal correlations and patterns.The output layer of the RNN commonly employs a linear activation function to generate continuous numerical predictions for regression applications like the prediction of wheat yield. To forecast the future yield of wheat, the model learns to map the previous time-series data, comprising multispectral data and historical yield values. Recurrent connections in the RNN enable it to capture the seasonality and dynamics in the wheat production data, which can be influenced by a variety of environmental factors, weather patterns, and farming practices. The RNN can produce more precise and context-aware predictions by taking into account the historical context and temporal patterns.
Long-term dependency learning, however, can be hindered by RNN restrictions such the disappearing and expanding gradient difficulties. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), more sophisticated variations that provide superior memory and gradient flow through the network, have been created to overcome these problems.

2.7.2.1. Long Short Term Memory

Long Short-Term Memory (LSTM) networks were introduced by Sepp Hochreiter and Jurgen Schmidhuber in 1997 as an extension of RNNs. LSTM networks are a type of RNNs that have the ability to learn long-term dependencies in sequential input. LSTMs use a complex memory mechanism to selectively store and discard information over time, in contrast to typical RNNs, which struggle to remember data from the previous time steps. This makes it possible for them to process input sequences of any length and identify complex patterns in the data. An important application of LSTMs is time series

analysis. Because LSTMs are able to capture long-term dependencies in sequential data, it makes sense to use them in grain yield prediction as well.
At the core of an LSTM network, is a memory cell that can store information over time. Three different types of gates, known as input, forget, and output gates, regulate how information enters and leaves the cell. A sigmoid activation function is used to implement each gate, and it produces values between 0 and 1 that indicate how open or closed the gate is. The input gate controls how much new information can enter the cell. The forget gate decides how much of the present cell state should be kept or thrown away. The output gate, in the end, chooses how much of the current cell state should be sent to the next layer.
There is another type of LSTM called Bidirectional LSTM (BiLSTM). It has the ability to look at the input sequence in both forward and backward directions. It has two sets of hidden layers that are each used for forward and backward direction. Overall, LSTMs and BiLSTMs are effective methods for handling sequential data, and they have been successfully used in applications such as speech recognition and language translation. They can also prove to be very useful in predicting the grain yield of a wheat field by taking sequential data into consideration. Figure 2 gives a general insight to how an LSTM Model works (C Olah, 2015) [12].

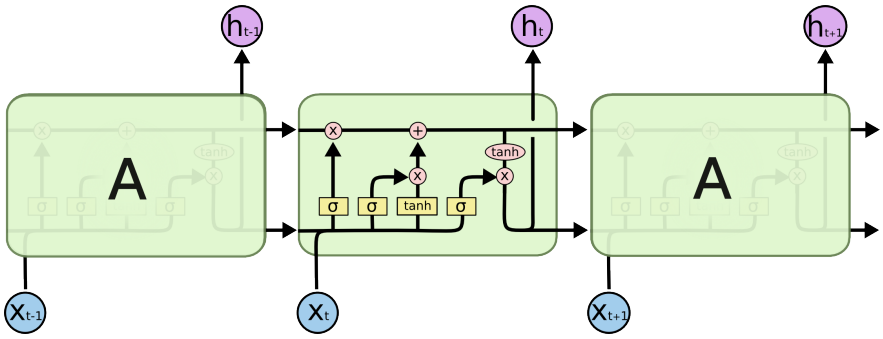


Fig. 2: General Diagram of a LSTM Model

The alpha symbol (α) represents the Forget Gate Activation. Specialized gates in the LSTM architecture regulate the information flow via the cell state. The forget gate is one of these gates, which chooses which information from the previous cell state should be kept or forgotten. The concatenation of the current input and the prior hidden state serves as the input for the forget gate activation, also referred to as ”alpha,” which is a sigmoid activation function. For each cell state element, it outputs a value between 0 and 1, indicating how much data should be remembered (1) or forgotten (0) for that element. The LSTM should retain the information if the value is close to 1, but it should forget the information if the value is close to 0.

Another essential element of the LSTM cell is the hyperbolic tangent activation function, denoted as ”tanh.” It is utilized in the update gate, input gate, and output gate, among other LSTM components. Tanh is a non-linear activation function that can handle both positive and negative data since it transfers the input to a value between -1 and 1. In the context of the LSTM, the tanh activation function is applied to the candidate cell state, which is a new candidate value that could be added to the cell state. It outputs a new candidate value between -1 and 1, concatenating the prior hidden state with the current input. How much of this candidate value will be used to update the current cell state is determined by the update gate (sigmoid activation). Overall, the LSTM cell is better able to capture long-range dependencies and solve the vanishing gradient problem that is frequently present in conventional recurrent neural networks, thanks to the combination of the forget gate activation (”alpha”) and the tanh activation function.

2.7.2.2. Gated Recurrent Unit

GRU networks were first introduced by 2014 as a simpler alternative to LSTMs (Cho, et al,2014) [13]. With fewer parameters and a gating mechanism similar to LSTMs, they are simpler to train and less prone to overfitting. The use of a reset gate and an update gate to regulate the information flow in the network is the main innovation of GRUs. Similar to LSTMs, this network can also selectively recall or forget information by using the gates to decide how much information from the previous time step should be transferred to the current time step. Backpropagation through time (BPTT), a version of backpropagation that can manage the temporal structure of sequential data, is often used to train GRU networks. The network's weights are changed during training in order to minimize a loss function that calculates the difference between the predicted and actual output. Like the LSTMs GRU network is also a powerful tool for modeling sequential data. In theory, the are more simpler and faster alternative to LSTMs. We can anticipate more advancements in the field of GRUs, as deep learning research progresses.

2.8. Hyperparameters

In contrast to learning from the data itself, hyperparameters are important factors in machine learning and deep learning models that control how the learning algorithm functions. Hyperparameters, which are defined by the user prior to training and have an impact on the model's behavior and performance, contrast with model parameters (sometimes referred to as trainable parameters), which are learned during training to suit the data. These hyperparameters and other relevant terminologies are defined as follows.

2.8.1. Trainable Parameters

The variables that a neural network learns during the training phase are referred to as trainable parameters in the context of deep learning. They are in charge of identifying trends and relationships in the data and represent the model's weights and biases. In order to reduce the discrepancy between the model's predictions and the actual target values, the model iteratively adjusts these trainable parameters throughout training. Through optimization algorithms like gradient descent or its derivatives, the parameters are adjusted. The effectiveness of machine learning models depends heavily on the use of trainable parameters. The model can generalize well and make precise predictions on fresh, untainted data by discovering the best values for these parameters from the training data.

2.8.2. Activation Functions

Activation functions play an important role because they determine the output of a neuron based on the input it receives. By bringing non-linearity to the neural networks, activation functions are essential to deep learning. They enable the network to understand intricate patterns and enable it to carry out a variety of tasks, including classification, regression, and others. Without non-linear activation functions, the network would operate as a single linear function, which would restrict its ability to recognize such patterns and connections in the data. The activation functions relevant to this study are briefly discussed in the following section.

2.8.2.1. Rectified Linear Unit (ReLU)

ReLU is one of the most prevalent activation methods in deep learning. ReLU facilitates effective gradient propagation, which reduces the vanishing gradient issue and expedites training. While improving on earlier functions, it also has a potential of 'dying' during training (Changhau, 2017) [14]. ReLu activation function is used in hidden layers because it is easy to compute and it can learn complex patterns in the data. It is a simple function that outputs the input unchanged if it is positive and 0 if it is negative. Mathematically, ReLU can be defined as:

$(x) = max(0, x)$

2.8.2.2. Linear Activation Function

The linear activation function is commonly employed in the output layer of the neural network for regression problems in deep learning. The linear activation function performs a direct linear mapping of the input to the output, in contrast to other activation functions which incorporate non-linearity. It is denoted as $f(x) = x$ where x is the input value this function receives and $f(x)$ is the output.

2.8.3. Loss Function

A loss function, often referred to as a cost function or an objective function in deep learning, measures the variance between the trained model's predictions and the actual target values. A loss function is an objective function that is being optimised during the learning process (Keras, 2023)[15]. The objective is to reduce the overall loss, which gauges the model's effectiveness. There are many loss functions for different purpose, but for this study Mean Squared Error (MSE) is preferred. In regression problems, the Mean Squared Error (MSE) loss function is often utilized. Between the projected values and the actual target values, it calculates the average squared difference. MSE is sensitive to outliers since it penalizes larger errors more harshly than smaller ones. The objective of training is to minimize the MSE, which essentially entails lowering the overall prediction error and raising the predictive accuracy of the model.

2.8.4. Epochs

In the context of deep learning, an epoch is a whole iteration through the entire training set. The model's weights are changed at the beginning of each epoch in accordance with the loss function and the optimization technique of choice. A model can learn from the data several times and increase its performance by training it via multiple epochs. The complexity of the job and the quantity of the dataset must be taken into account while tuning the number of epochs, which is a hyperparameter.

2.8.5. Batch Size

The amount of training samples utilized in each iteration of the optimization method during training is referred to as the batch size. The model adjusts its weights based on the average gradient calculated over the batch after batching the training data. Larger batch sizes demand more memory but can speed up training because more samples can

be processed concurrently. While training may take longer with smaller batch sizes, generalization and convergence may be improved.

2.8.6. Dropout Layer

The dropout layer is is employed in neural networks and the main purpose of it is to perform regularization and avoid overfitting. A portion of the layer's neurons are randomly dropped out or silenced during training. Due to its inability to significantly rely on a small number of neurons, this encourages the network to acquire more reliable representations. Dropout contributes to the ensemble effect, where various neuronal subsets are active during training, improving generalization and improving predictions on unobserved input. Dropout reduces complex co-adaptions of neurons by denying neurons the option of relying on the presence of other neurons (Alex Krizhevsky, et al, 2012)[16].

2.9. Model Accuracy Metrics

Model accuracy metrics are used to evaluate the performance of machine learning models. There are many different metrics available depending on the problem and dataset, but for this study two popular metrics for regression are used.

2.9.1. Coefficient of Determination

Coefficient of Determination also denoted as R^2 is an accuracy metric for regression problems. It calculates the percentage of the dependent variable's variation that can be predicted from the independent variables. The range of R^2 values is 0 to 1, with 1 denoting the best fit and 0 denoting no association between the variables. R^2 values that are closer to 1 suggest that a bigger amount of the variation is captured by the model, which indicates improved prediction accuracy.

2.9.2. Root Mean Squared Error

Root Mean Squared Error (RMSE) is another popular accuracy metric for regression problems. It calculates the average difference between the dependent variable's expected value and its actual value. Lower values denote better performance of the model. RMSE

values vary from 0 to infinity. RMSE can be written mathematically as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2}$$

where n is the number of data points or predictions, $y_{\text{true},i}$ is the actual value of the i-th data point, and $y_{\text{pred},i}$ is the predicted value for the i-th data point. Both R^2 and RMSE are valuable metrics in assessing the accuracy and performance of regression models. R^2 provides an indication of the model's explanatory power, while RMSE gives insight into the magnitude of prediction errors. It is important to consider both metrics together to evaluate the model comprehensively. High R^2 and low RMSE values suggest a well-performing model with good predictive accuracy.

2.9.3. Overfitting and Underfitting

Sometimes a machine learning model performs exceptionally well on training data but poorly on untried or fresh data. In such case, however, the model exhibits a high bias and the situation is classified as underfitting (Rashcka, et al, 2019) [17]. In other words, rather than discovering underlying patterns, the model memorizes the training data. As a result, it performs poorly on new data and captures noise and random fluctuations in the training set. When the model is very complicated or the training data is insufficient, overfitting is more likely to happen, causing the model to learn from noise rather of interesting patterns. Numerous methods, like using more training data, fewer layers or neurons, adding regularization (such dropout), or early termination during training, can be used to solve overfitting.

Underfitting, on the other hand, occurs when a machine learning model is too basic to capture the fundamental patterns in the data, leading to subpar performance on both the training data and fresh data. The model's low accuracy and high error rates are caused by its inability to understand the key characteristics and connections in the data. Low training and validation/test accuracies or losses, as well as its difficulty in learning even the training data, are indications of underfitting. Underfitting happens when the data is either too complex or noisy for the model to handle, or when the model is not complex enough to reflect the data. The best approach is finding the sweet spot between underfitting and overfitting, where the model generalizes effectively to new data while capturing the pertinent patterns from the training data.

3. METHODOLOGY

3.1. Schematic of The Approach

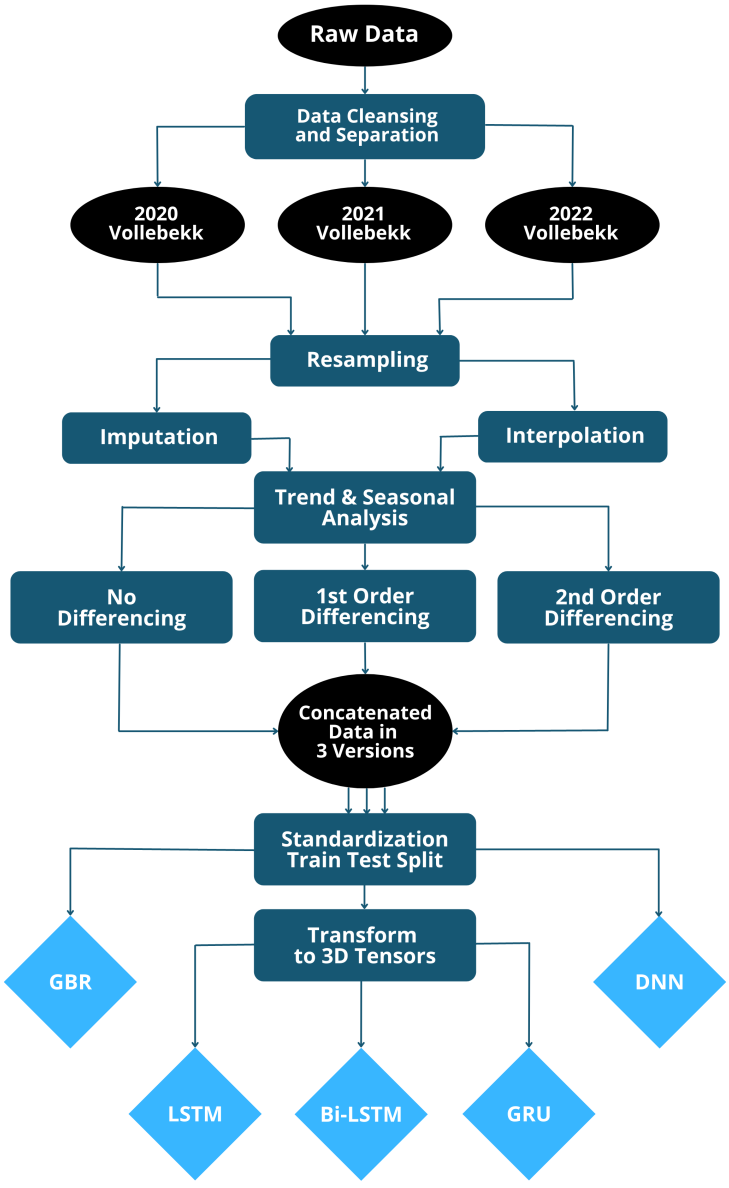


Fig. 3: Schematic of Approach to the work

The workflow for this study is shown in Figure 3. The features of interest were selected from the raw dataset and that specific section of the dataset was then separated into three subsets with respect to each year, that is 2020, 2021 and 2022. The selected features can be seen in Table A.2. The purpose of separating the data was for the ease in management and pre-processing. Each set was processed separately. After going through

resampling, a copy of each of these sets were made to be filled by using Imputation and Interpolation separately. The purpose of doing this was to compare the performance of models using different approaches. Similarly more copies of all these instances were processed separetly while performing 1st order differencing, 2nd order differencing or no differencing at all. So for the whole study, around 6 instances or copies of all the 3 subsets of the data up until pre-processing. These subsets were concatenated at the end to form one clean and processed dataset for the models. A brief description of all the processes is given in Appendix A.3.

3.2. Data Collection

In the vPheno project, to make data collection automated, a fairly popular drone DJI Phantom 4 (DJI, 2016)[18] was used to collect data throughout the years. This UAV was equipped initially with MicaSense RedEdgeM camera that could take multi-spectral images with a wavelength of upto 840nm (MicaSense, 2017) [19]. But with the rise in popularity DJI decided to make their own multi-spectral cameras, named as P4M in this dataset. From 2021 P4M was used as the main camera for data collection. The images were captured using a total of 5 bands, namely Red, Green, Blue, Near Infrared (NIR) and RedEdge. The wavelength of these bands can be seen in Table 1. The UAV is also equipped with a Downwelling Light Sensor. It helps with the calibration of images when there is a change in lighting conditions. With the help of software applications such as Pix4D (Pix4D SA, Lausanne, Switzerland))[20] the drone takes a pre-defined flight path, captures the images and then stitches it all together. The numerical reflectance values across all 5 bands and all the plots in the field, are then stored in an excel file. The data was collected at two farms, namely Vollebekk and Staur. Multiple traits such as Days to Maturity, Days to Heading, Grain Yield and many more were observed and collected in the dataset for all the different conducted studies. The collection started from June until August every spring season from 2019 to 2022. In 2019, the data was collected at 8 different and irregular time points throughout the season. While in 2020, 2021 and 2022 there were 12, 22 and 23 observations per plot respectively. Each plot has a unique identification number. Because the frequency was inconsistent, it made the problem more challenging, which means a lot of pre-processing needed to be done.

Band		Wavelength
Blue	-	475
Green	-	560
Red	-	668
RedEdge	-	717
Near-Infrared (NIR)	-	840

Table 1: Wavelength of Bands

3.3. Computational Resource

All computations were performed using a personal computer equipped with Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 7.86GB of useable DDR3 Random Access Memory and solid state drive. Windows 11 Pro Version 22H2 was the operating system. From pre-processing to predictions using machine learning and deep learning models and general programming related stuff, Python 3.9.12 in combination with Jupyter Notebook IDE were used. Table 2 gives some information about the important libraries used for this study alongwith their purpose.

Library	Purpose
Pandas	Provides data structure functions and helps in reading data from csv file and storing in the memory in the form of a dataset.
Numpy	The fundamental library for all sort of numerical computing. Helps with the handling of large and multi-dimensional arrays.
Matplotlib	Provides data visualization tools to create graphs, charts, figures and plots.
Scipy	An extension of Numpy library that can perform interpolation, imputation and statistical functions.
ScikitLearn	Provides efficent tools for data analysis alongwith machine learning algorithms for classification, regression and clustering, etc.
TensorFlow	A deep learning framework that contains many tools to create machine learning and deep learning models.

Table 2: Python Libraries used for this study

3.4. Pre-Processing Steps

As mentioned in the previous section, the data was collected at inconsistent time points throughout these three years of interest. For proper analysis and to give all the models a fair chance to prove their worth, a time series data with consistent and equal time steps was absolutely necessary. All the procedures were separately applied to each year's data and then they were concatenated to form one big dataset. The data had to go through the following pre-processing steps.

3.4.1. Addressing the inconsistent frequency

To bring all the observations at the same frequency, a function called 'resample' by Pandas library was used. It resampled each year's data to a frequency of 5 days, which in other words mean that the data was now arranged in such a way that only the rows with a time difference of 5 days were kept and the rest were automatically removed by this function. This of course led to some null values as well, at the days where data was not originally captured. In each year, these rows ranged from 3 to 4 missing values per plot. But the rest of the time points in each year were more than enough to fill their place by either using Imputation or Interpolation. Multiple approaches were tested since the data did not show any linear relation within itself. After experimenting, one copy of the data was filled by imputation and another copy of the same data was filled using cubic spline interpolation.
The imputation was done by using 'fillna' function of Pandas, filling the empty rows by the median of the rest of the values in that specific column. While the interpolation was achieved using a library called 'Scipy'. The function that was used is known as 'CubicSpline'.

3.4.2. The Time Series Components

Before we could merge all 3 years in a single dataframe, we needed to do some time series analysis on each year to figure out if there is something which can affect or improve the models' performances. The first step was to check that the data was stationary because a non-stationary data can affect the performance of some models such as LSTM. The p-value for each feature was calculated using 'adfuller' function from statsmodels library. Luckily the p-value for all the features was less than 0.05 which meant that the data was stationary. The next step was to determine if the data had any trend or seasonality that needed to be taken care of. It can be done by decomposing the time series data

into its trend, seasonal and residual components using 'seasonal_decompose' function of 'statsmodels' library. Each of the 5 features for each year needed to be decomposed separately. And every feature showed a trend but no seasonality. This problem was addressed by 'differencing' which is just literally taking a difference among the subsequent observations.

3.4.3. Outlier Removal

After the intial pre-processing phase, each year had 10 observations per plot across similar time points and frequency. Multiple thresholds for IQR Percentile were tested. Lower threshold of 20 and upper threshold of 80 produced the best results. This was easily achieved by using 'percentile' function from the Numpy library.

3.4.4. Standardizing and Splitting the Data

The concatenated dataset was standardized using 'Standard Scaler' class and its function 'fit_transform' from Scikit Learn library. For splitting the data into train, test and validation set, again Scikit Learn's 'Model Selection' class was used for its function 'train_test_split'. 20 percent of the input and output variables were used for evauating the model. The rest of the 80 percent of training data was again split into 80 to 20 ratio. The 20 percent of which was used as a validation set.

3.5. Model Summary

Multiple models with different hyperparameters were trained and tested. After a lot of trial and error, it was decided to only include the following models for this study. This subsection provides an overview of the structure of these models.

3.5.1. Gradient Boosting Regressor

For this machine learning model, even after some trial and error, the default hyperparameters were used and 0.1 was the learning rate. The criterion for measuring the quality of the split was a special case of mean squared error called Friedman MSE.

3.5.2. Dense Neural Network

To choose the best performing DNN Model, multiple hyperparameters were tested. Table 3 shows the structure of a rather dense model but simpler versions were also tested,

however this model came on top in terms of performance as compared to the others. It consisted of six hidden layers with decreasing units or nodes in each layer. As mentioned in the previous sections and also concluded by testing the model, ReLU activation function performs the best for the regression task in this study. Linear activation function in the output layers gets the model a total of 140,737 trainable parameters. Mean Squared Error was used as the loss function and Adams as the optimizer. This remains constant for all the other models as well. The model was set to train for 400 Epochs to give it the proper time it required to learn from the data. The batch size for this model was 128.

Layers	Units	Activation Function
Dense	256	ReLU
Dense	256	ReLU
Dense	128	ReLU
Dense	128	ReLU
Dense	64	ReLU
Dense	64	ReLU
Dense	1	Linear

Table 3: Structure of DNN Model

3.5.3. Long Short Term Memory

The structure of LSTM model as shown in Table 4 is similar to DNN, but the noticable difference is that instead of the usual dense layers, it now has LSTM layers. ReLU and Linear were used as the activation funtions for hidden layers and output layer respectively. This model also trained for 400 epochs with a batch size of 64. The total trainable parameters for this model were 527,265.

Layers	Units	Activation Function
LSTM	256	ReLU
LSTM	128	ReLU
LSTM	64	ReLU
LSTM	32	ReLU
Dense	1	Linear

Table 4: Structure of LSTM Model

3.5.4. Bidirectional LSTM

A rather simpler version of Bi-Directional LSTM was taken into consideration which will be further discussed in the next section. Table 5 shows that only two bidirectional

layers were used, giving it a number of 134,785 in trainable parameters. Batch size of 350 was used, and this model was trained for 500 Epochs.

Layers		Units		Activation Function
Bidirectional	-	64	-	Relu
Bidirectional	-	64	-	Relu
Dense	-	1	-	Linear

Table 5: Structure of Bidirectional LSTM

3.5.5. Bidirectional LSTM with Dropout

Table 6 shows another variation of Bidirectional LSTM that was taken into consideration to observe the effect of Dropout Layers in the model's performance for this context. A dropout rate of 0.2 means that during each training iteration, approximately 20% of the neurons in a specific layer will be randomly set to zero, hence giving it a chance to be more regularized. This model was also trained for 500 epochs and with a batch size of 350.

Layers		Units		Activation Function
Bidirectional	-	64	-	ReLu
Dropout	-	0.2	-	-
Bidirectional	-	64	-	ReLu
Dropout	-	0.2	-	-
Dense	-	1	-	Linear

Table 6: Structure of Bi-LSTM Model with Dropout

3.5.6. Gated Recurrent Unit

Another type of RNN was employed to compare it with the previous models. Table 7 shows the structure of the GRU Model with different layers such as batch normalization and dropout. It had a total of 98,977 trainable parameters and it was tested for 400 Epochs with a batch size of 128. Although a simpler version of this model was also tested but both gave similar results, more about it in the next section.

Layers		Units		Activation Function
GRU	-	128	-	ReLu
Batch Normalization	-	-	-	-
Dropout	-	0.2	-	-
GRU	-	64	-	ReLu
Batch Normalization	-	-	-	-
Dropout	-	0.2	-	-
GRU	-	32	-	ReLu
Batch Normalization	-	-	-	-
Dropout	-	0.2	-	-
Dense	-	1	-	Linear

Table 7: Structure of GRU Model with Dropout

4. RESULTS AND DISCUSSION

4.1. Data Preparation

4.1.1. Raw Data and Selected Data

The data was collected at Vollebekk and Staur Farm as mentioned earlier, but for this study only Vollebekk data from 2020 to 2022 was used because 2019 data had not enough time points. In its raw form, it had 24 columns that were merely used as a reference of the location of the plot and some other factors that were not suitable to be treated as a feature. The total number of observations were 57,264 which included both farms and early observations from 2019. Figure 4 shows the structure of raw data. These columns were helpful in initial data preparations, sorting and separating them with respect to year and location. The features of interest can be seen in Table A.2. The data was divided into three separate datasets, one for each year's observations. This helped with the further pre-processing steps and made it easier to manage. After all the steps were applied, these three years were concatenated to make one single dataset again. Only the plots marked as the type 'yield' were selected. For each yield plot, 10 time points from each year were taken into consideration. Since there were a total of 5 bands, it made a total of 50 columns or features. To train a model, the data needs to be in a wide format which explains why the number of columns in the final data are more than that of the raw data. The selected data at the end consisted of 1435 observations and 50 features. For creating training set, test set and validation set, this dataset was split into 918, 287 and 230 rows or observations respectively.

season	location	plot_number	plot_type	camera	line_number	masbasis2015	line	rep	...	blue_median	green_median	red_median	rededge_median	nir_median
2022	vollebekk	1892.0	biomass	p4m	1327.0	1327.0	NaN	3.0	...	0.056673	0.095848	0.117809	0.153384	0.176198
2022	vollebekk	1892.0	biomass	p4m	1327.0	1327.0	NaN	3.0	...	0.056812	0.095473	0.114176	0.147188	0.177045
2022	vollebekk	1892.0	biomass	p4m	1327.0	1327.0	NaN	3.0	...	0.051293	0.084907	0.102908	0.143419	0.167718
2022	vollebekk	1892.0	biomass	p4m	1327.0	1327.0	NaN	3.0	...	0.034919	0.055395	0.065450	0.096945	0.115780
2022	vollebekk	1892.0	biomass	p4m	1327.0	1327.0	NaN	3.0	...	0.070500	0.116883	0.136900	0.193303	0.231470
2022	vollebekk	1892.0	biomass	p4m	1327.0	1327.0	NaN	3.0	...	0.056069	0.089795	0.111481	0.156016	0.187142

Fig. 4: A few rows from the raw data

Target variables stored in a separate file had many traits observed for similar other studies. Of all the traits, predicting the Grain Yield is the main focus of this study. Figure 5 shows the target variables data in its raw form, where grain yield is denoted by 'GY_g_m2' and the units used are grams per meter square (g/m^2). The plot number from both the files were used as an index to later concatenate both files to each other.

	env	season	location	plot_number	plot_type	line_number	masbasis2015	rep	block	column	DH_dss	DM_dss	GY_g_m2	PH_cm	GPC_pct
3801	2020_vollebekk	2020	vollebekk	1105	yield	6.0	1029.0	1.0	1.0	5.0	66.0	118.0	713.333333	81.67	10.4
3802	2020_vollebekk	2020	vollebekk	1106	yield	1543.0	1543.0	1.0	1.0	6.0	68.0	119.0	677.333333	86.67	12.1
3803	2020_vollebekk	2020	vollebekk	1107	yield	1338.0	1338.0	1.0	1.0	7.0	66.0	120.0	361.333333	95.00	15.5
3804	2020_vollebekk	2020	vollebekk	1108	yield	1526.0	1526.0	1.0	1.0	8.0	67.0	117.0	697.333333	86.67	10.8
3805	2020_vollebekk	2020	vollebekk	1109	yield	1614.0	1614.0	1.0	1.0	9.0	68.0	118.0	664.000000	76.67	11.5

Fig. 5: A few rows from the target variables data

4.1.2. Resampled and Filled Data

Since each year had different number of observations, some rows were manually removed to bring all the three years to a equivalent time frame and exactly the same number of observations. The initial observations where the plant was still in the 'heading' phase were the best rows to remove. This information was available in the target variable file. In Figure 6 it can be seen that the dates at which data was collected were inconsistent.

plot_number	env	season	location	plot_type	camera	lodging	date	blue_median	green_median	red_median	rededge_median	nir_median
1105.0	2020_vollebekk	2020	vollebekk	yield	mica	False	2020-06-18	0.015137	0.037526	0.014911	0.109639	0.431767
1105.0	2020_vollebekk	2020	vollebekk	yield	mica	False	2020-06-24	0.014972	0.034236	0.016068	0.094503	0.391461
1105.0	2020_vollebekk	2020	vollebekk	yield	mica	False	2020-06-26	0.012624	0.027086	0.015051	0.075851	0.435237
1105.0	2020_vollebekk	2020	vollebekk	yield	mica	False	2020-07-01	0.017417	0.043442	0.020844	0.107552	0.433779
1105.0	2020_vollebekk	2020	vollebekk	yield	mica	False	2020-07-08	0.022297	0.048837	0.020287	0.114728	0.444705

Fig. 6: Inconsistent Time Points

The data had a consistent frequency after Resampling the data. But it led to some null values as shown in Figure 7 because of the obvious fact that the data was not recorded at

that specific time in real life. This was dealt with two separate approaches. Two copies of the data were made, one was filled using Interpolation and the other was filled with Imputation to compare the results of the models.

plot_number	date	blue_median	green_median	red_median	rededge_median	nir_median
1105.0	2020-06-18	0.015137	0.037526	0.014911	0.109639	0.431767
1105.0	2020-06-23	0.013798	0.030661	0.015559	0.085177	0.413349
1105.0	2020-06-28	0.017417	0.043442	0.020844	0.107552	0.433779
1105.0	2020-07-03	NaN	NaN	NaN	NaN	NaN
1105.0	2020-07-08	0.022297	0.048837	0.020287	0.114728	0.444705
1105.0	2020-07-13	0.011046	0.031304	0.016580	0.070497	0.232059
1105.0	2020-07-18	0.022397	0.060377	0.048461	0.138072	0.290974
1105.0	2020-07-23	NaN	NaN	NaN	NaN	NaN
1105.0	2020-07-28	0.027139	0.075271	0.070734	0.173504	0.311046
1105.0	2020-08-02	NaN	NaN	NaN	NaN	NaN
1105.0	2020-08-07	0.031707	0.071970	0.110121	0.155774	0.253116
1105.0	2020-08-12	0.042600	0.077703	0.126265	0.151689	0.252579

Fig. 7: Resampled Data causes Null Values

4.1.3. Differencing the Data

Since each wheat plot was observed at multiple time points, the seasonal decomposition was applied to each plot separately because each could be identified as a variable of its own. However because of similar conditions all across the plots marked as yield in the dataset, the time series components were also similar. Figure 8 shows the trend, seasonal and residual components of one of the plots across all three years, marked as plot 7 as an internal reference for the code.

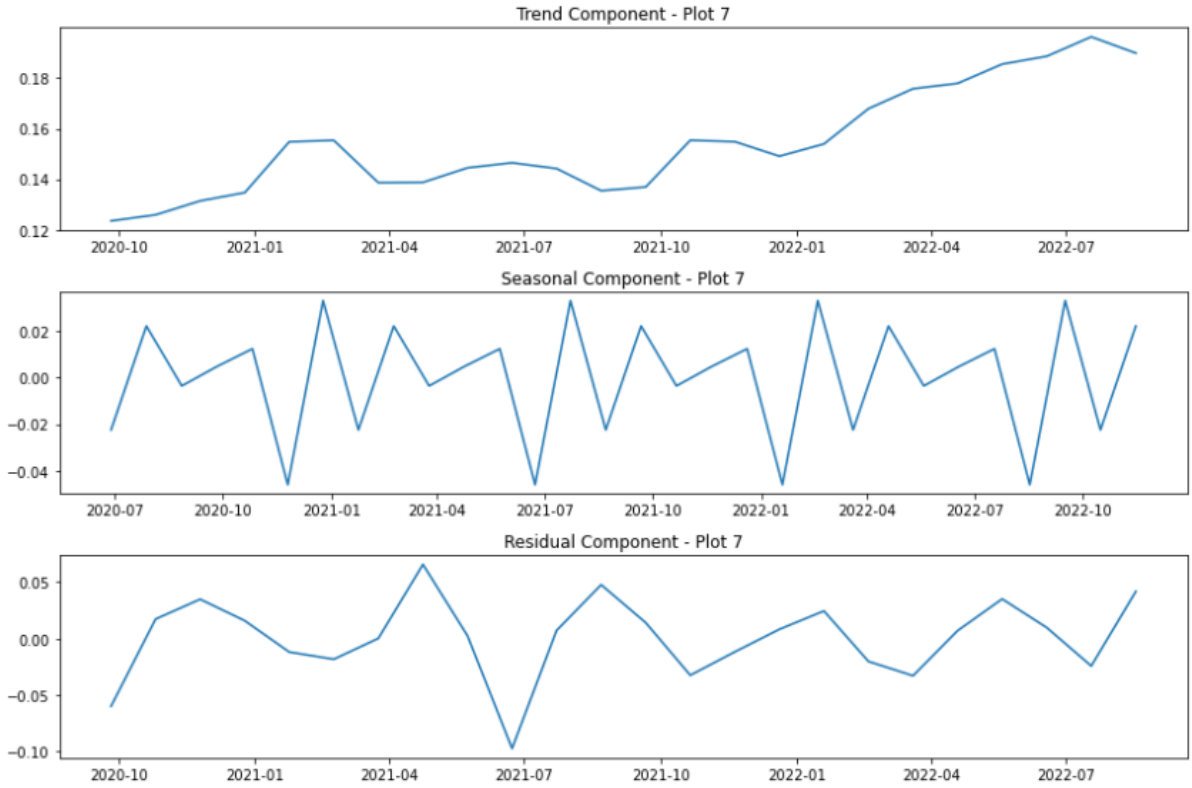


Fig. 8: Trend, Seasonal and Residual Components of one plot

It can be seen that the data has a clear rise in the trend component. Also visually inspecting the seasonal component verifies that there is a seasonal factor also present in the data. This calls for differencing the data to minimize the effect of these components on the accuracy of the models. There is a noticable uncertainty in the residual component as well which shows that there may be possible anomalies and outliers present in the data. Multiple approaches were once again used while differencing to compare the results of the models. After some trial and error, data with first order differencing and second order differencing were tested separately on the best performing models. A copy of data without any differencing was also used to identify if the models can handle such data

without a lot of pre-processing.

4.1.4. Correlation Plot

Upon inspection of the correlation plot using all 5 features, it can be seen that the some features have high correlation with each other as shown in Figure 9. At this point two approaches were used. Once again two copies of the processed data were made, one was used as it is and in the other one, two columns namely blue_median and red_median were removed and tested separately on the models. The purpose of this approach was once again comparing the results to see which one leads to a better performance of the model. Blue and red bands were removed because of a couple of reasons. They showed high correlation with other features and in a wheat field, these colors are less prominent in theory.

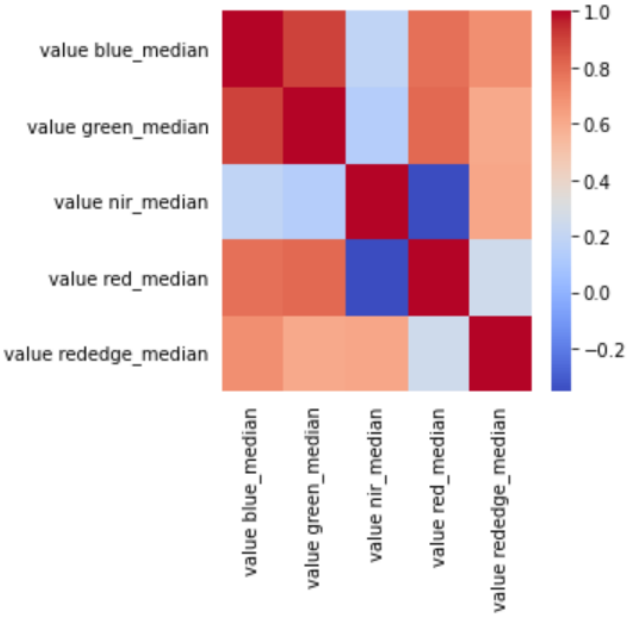


Fig. 9: Correlation Plot among all features

4.2. Evaluation of the Models

4.2.1. Loss Plot

Any model at its first epoch will have a huge error, and then it slightly drops across more epochs. So the DNN model used for this study also shows the same trend as shown in Figure 10. When noticed closely, the loss error keeps drops slightly even after 300

epochs. This is the reason why these models were given at least 400 epochs to get to the optimum spot of having the least amount of error but also avoiding overfitting.

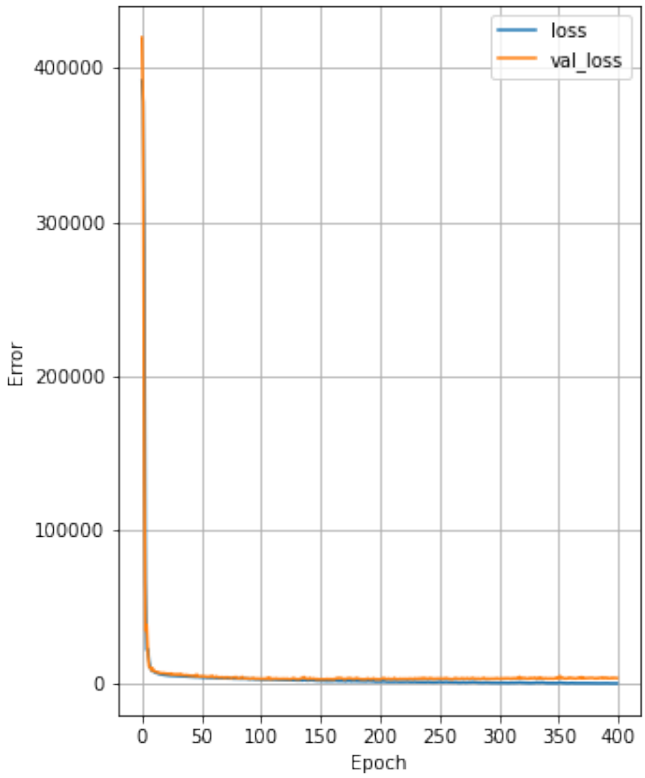


Fig. 10: Loss Plot of DNN Model

4.2.2. Discarded Models

To create a starting point for competing the models with each other all models were trained on imputed data without applying any differencing. As it can be seen in Figure 11 the LSTM Model did not learn anything at all. Similar situation was achieved by using GRU. So from this point, these two models were discarded due to being practically useless for further comparison. Later on after further testing, Bidirectional LSTM with Dropout layers was also discarded, which is the reason why the test score and RMSE for test data was not recorded at all for these models. Only the three best performing models were taken into consideration for the rest of the study. As far as this specific approach is concerned, GBR gave the best overall performance with 0.87 and 0.71 of train and test score respectively. But this will be addressed in the next subsection. An RMSE of 35.23 and 52.62 can also be observed in Figure 12. It should also be kept in mind that the units for grain yield are g/m^2, and the RMSE value is with respect to that range.

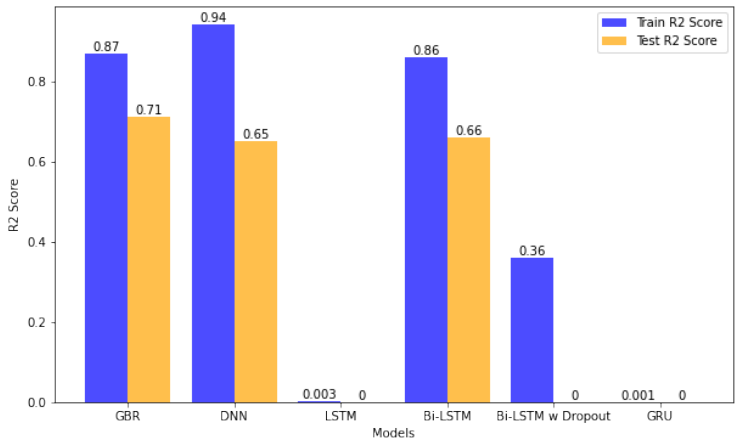


Fig. 11: R^2 - Imputed Data and No Differencing

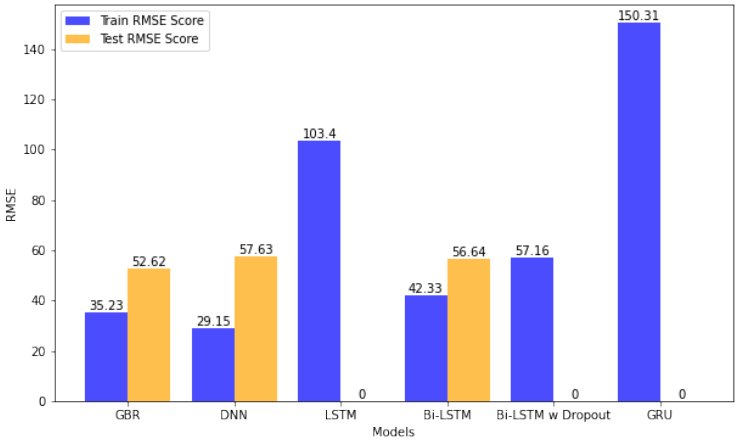


Fig. 12: RMSE - Imputed Data and No Differencing

4.2.3. Effect of Differencing on Overfitting

When the same imputed data is gone through first order differencing, the following results are achieved as shown in Figure 13 and Figure 14. It can be seen that the performance of GBR drops on both train and test data. DNN on the other hand has a drop in its training accuracy, however the test accuracy is almost the same. This is indicates that the models were overfitting previously. Instead of learning from the data, they were memorizing it. However, on the contrary, the perform of Bidirectional LSTM slightly increases after differencing. This proves that Bidirectional LSTM is taking the sequence

into consideration, and after the effect of trend and seasonality is reduced, it gives more accurate results because the data is now more stationary.

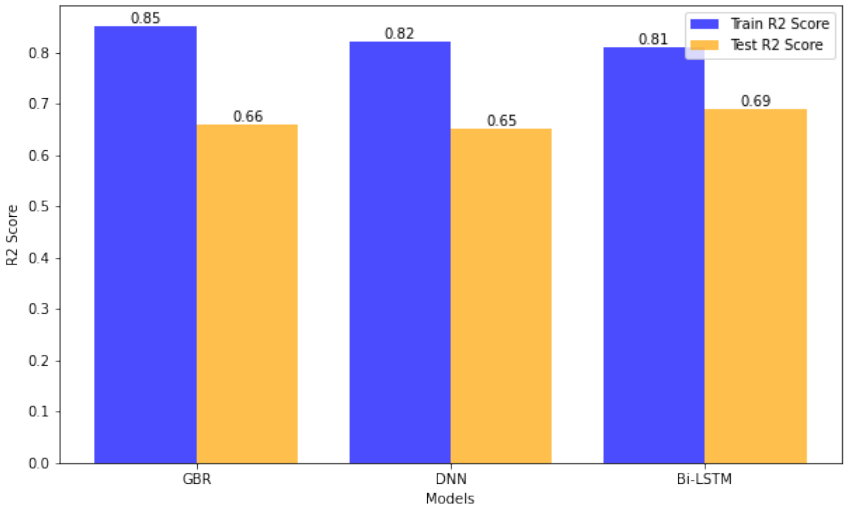


Fig. 13: R^2 - Imputed Data and First Order Differencing

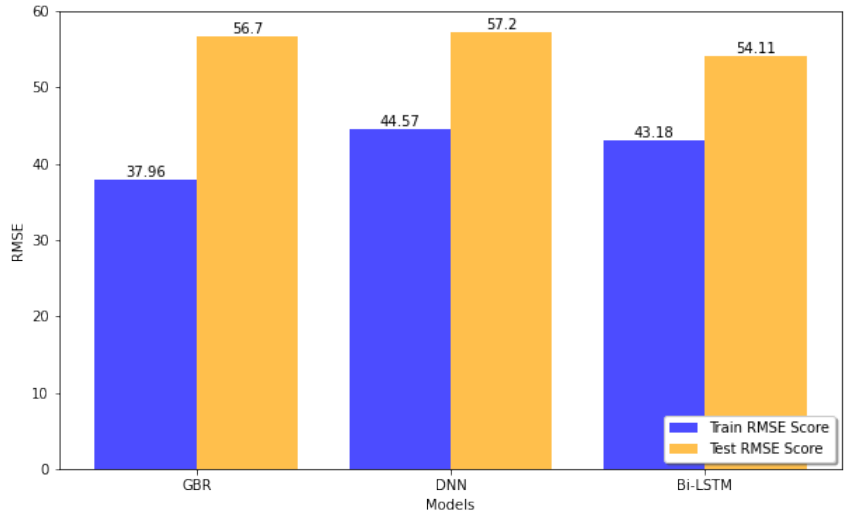


Fig. 14: RMSE - Imputed Data and First Order Differencing

4.3. Best Performing Approach

As discussed in the previous subsection, differencing the data helps with reducing the effects of trend and seasonality, makes the data more stationary. When second order

differencing is applied to the imputed data, it further reduces it. The performance of the models show the following results as shown in Figure 15 and Figure 16. A similar pattern for GBR can be seen, it no longer overfits but the performance slightly drops after each differencing. DNN's train and test score also have a less gap which means differencing also makes it less prone to overfitting. But the most balanced model with consistent train and test score is Bidirectional LSTM. The RMSE of these models also show similar patterns. So after a lot of trial and error, when Imputation was used to fill the empty spots in the data and then second order differencing was applied, it gave the best results. The results for interpolated data can also be seen in Appendix 4.

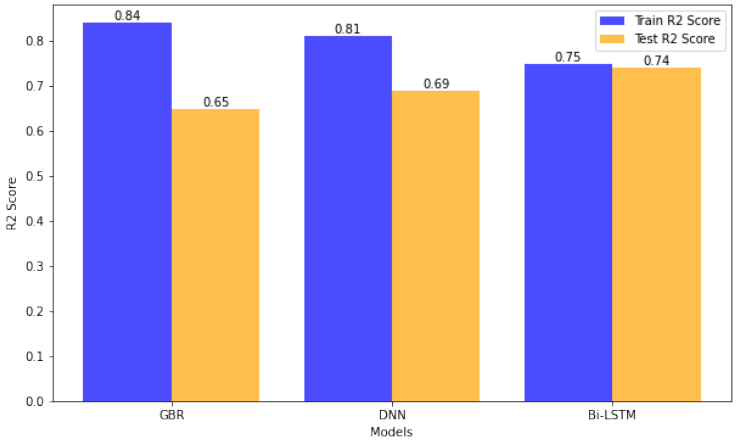


Fig. 15: R^2 - Imputed Data and Second Order Differencing

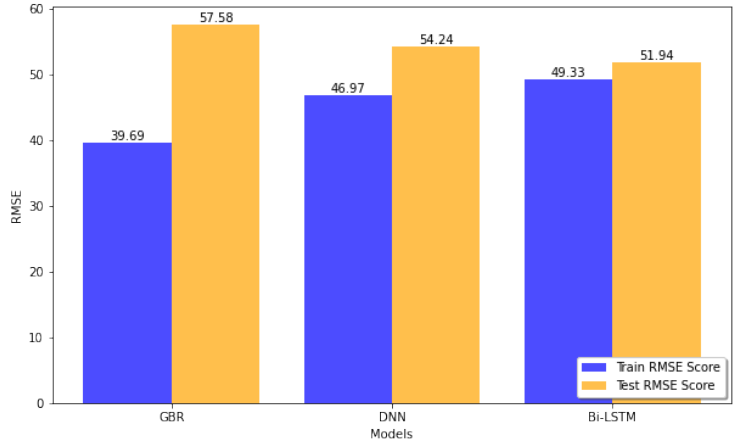


Fig. 16: RMSE - Imputed Data and Second Order Differencing

5. CONCLUSION

Using advanced deep learning techniques and a lot of multispectral time series data collected by unmanned aerial vehicles (UAVs), a significant endeavor was undertook in this thesis to forecast the grain yield of the wheat field at vollebekk. The goal was to contribute in the agricultural sector by creating precise and effective predictive models that may help farmers and decision-makers manage crop production and maintain food security. The accuracy and completeness of the data were crucial to the performance of our predictive models. Resampling for maintaining consistent frequency throughout the data in combination with interpolation and imputation to fill in the missing values of the data assured this robustness. The best outcomes were obtained using imputation and second-order differencing, underlining the significance of careful data pre-processing in getting the optimal model performance. Various deep learning and machine learning models were investigated, such as the Bidirectional Long Short-Term Memory (BiLSTM) networks, Deep Neural Networks (DNNs), and Gradient Boosting Regressor (GBR). On the basis of the dataset on wheat grain yield, BiLSTM stood out among these models as the most promising contender. BiLSTM was the best option for this work because of its capacity to capture temporal dependencies in the multispectral time series data, including the sequential character of the observations. The trend and seasonal component of the data caused GBR and DNN models to overfit, but it was later addressed by second order differencing.

The study helped to demonstrate the potential of multispectral UAV data to revolutionize agricultural methods and shed light on the possibilities of contemporary technologies. High accuracy grain production prediction opens the door to improved crop management, effective resource management, and sustainable agriculture practices. A number of difficulties were faced throughout the research, including choosing the proper hyperparameters, dealing with overfitting issues, and handling missing values. But the main difficulty was to bringing the observations of all the years to somewhat same time points. This included manual removal of the early observations when the crop was still not in the heading phase. This further helped in resampling. The models performance is admirable, however it is well trained on only one type of field. The dynamic nature of agricultural data and the significance of ongoing improvement and adaption to changing situations can be acknowledged. Future studies could examine strategies that combine domain expertise and outside data sources to produce predictions that are even more reliable and generalized. In conclusion, this study has laid the foundation for a new era of agricultural forecasting, where deep learning and multispectral UAV data converge to

drive informed decision-making and sustainable farming practices. And these findings will hopefully stimulate additional research in this area and help the international effort to ensure a resilient and food-secure future for upcoming generations.

Bibliography

[1] Jason Hill David Tilman, Christian Balzer and Belinda L. Befort, "Global food demand and the sustainable intensification of agriculture," vol. 108, pp. 20260–20264, 2011.

[2] Nathaniel D Mueller Paul C West Deepak K Ray, Navin Ramankutty and Jonathan A Foley, "Recent patterns of crop yield growth and stagnation," vol. 3, 2012.

[3] Massimo Minervini and Sotirios A. Tsaftaris, "Application-aware image compression for low cost and distributed plant phenotyping," pp. 1–6, 2013.

[4] Humboldt State University, "Spectral reflectance," 2018.

[5] Sebastian Raschka and Vahid Mirjalili, *Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*, Packt Publishing, 2019.

[6] Rob J Hyndman and George Athanasopoulos, *Forecasting: Principles and Practice*, OTexts, 2018.

[7] J. Yang S. E. Lee C. Deb, F. Zhang and K. W. Shah, *'A review on time series forecasting techniques for building energy consumption,' Renewable and Sustainable Energy Reviews, vol. 74*, 2017.

[8] Christopher M Bishop, *Pattern recognition and machine learning*, 2006.

[9] Muhammad Fahad Ijaz, "Spring wheat trait prediction using combined multi-environment, weather and multi-spectral timeseries uav data," 2021.

[10] F. Chollet, *Deep Learning with Python, 1st*, USA: Manning Publications Co., 2017.

[11] Micheal Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[12] Christopher Olah, "Understanding lstm networks," 2015.

[13] Caglar Gulcehre Dzmitry Bahdanau Fethi Bougares Holger Schwenk Yoshua Bengio Kyunghyun Cho, Bart van Merrienboer, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[14] Isaac Changhau, "Activation functions in neural networks," 2017.

[15] "Keras documentation: Losses," .

[16] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton., "Imagenet classification with deep convolutional neural networks.," *Advances in Neural Information Processing Systems*, 2012.

[17] S. Raschka and V. Mirjalili, *Python Machine Learning, 3rd Ed. Birmingham, UK*, Packt Publishing, 2019.

[18] "Phantom 4 pro - product information - dji," 2016.

[19] MicaSense, "Redegde-m user manual (pdf) - micasense knowledge base," 10 2017.

[20] "Pix4dmapper, lausanne, switzerland," 2017.

Appendix A

1. LIST OF ACRONYMS

Acronym	Full Form
ANN	Artificial Neural Network
Bi-LSTM	Bidirectional LSTM
DL	Deep Learning
DNN	Deep Neural Network
GBR	Gradient Boosting Regressor
GRU	Gated Recurrent Unit
GY	Grain Yield
LSTM	Long Short-Term Memory
ML	Machine Learning
NIR	Near-Infrared
R2	Co-efficient of Determinant (R^2)
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
UAV	Unmannaed Aerial Vehicle

Table A.1: Table of Acronyms

2. FEATURES OF INTEREST

Feature	Type	Brief Description
Plot Number	Index	A unique identifier for each wheat plot, that is also being used as an index
Date	Date Format	Has a record of the date at which each observation was made
Grain Yield	Numerical	The target variable that will be predicted by the models
Blue Median	Numerical	Median value of the reflectance of the blue band.
Green Median	Numerical	Median value of the reflectance of the green band.
Red Median	Numerical	Median value of the reflectance of the red band.
NIR Median	Numerical	Median value of the reflectance of the NIR band.
RedEdge Median	Numerical	Median value of the reflectance of the red edge band.

Table A.2: Features of Interest

3. DESCRIPTION OF WORK SCHEMATIC

Method	Description
Data Cleaning and Separation	Selecting only the features useful for training the model, removing any null values. Separating the data into 3 sets with respect to year.
Resampling	Bringing all the sets of the data to the same frequency
Imputation & Interpolation	Making a copy of all the sets and filling the empty values using both approaches.
Trend & Seasonal Analysis	Observing and analyzing any possible trend and seasonality component in each dataset.
No Differencing	Passing on the data for further processes without applying any differencing.
1st Order Differencing	Another copy of datasets is made. Differencing is applied once on all datasets in order to minimize the effect of trend and seasonal components.
2nd Order Differencing	Differencing is applied twice on a new instance of the datasets in order to compare which approach gives the best accuracy on the models.
Standardization	The three sets of the data are now concatenated in a single dataset. To bring the values on a unit variance standardization is applied.
Train Test Split	The concatenated dataset is now split randomly into train, test and validation set.
Transform to 3D Tensors	LSTM, Bi-LSTM and GRU require the data to be in the form of 3D Tensors, so this transformation is applied before training these models.

Table A.3: Applied methods and their purpose

.

4. PERFORMANCE OF ALL MODELS

4.1. Imputed Data

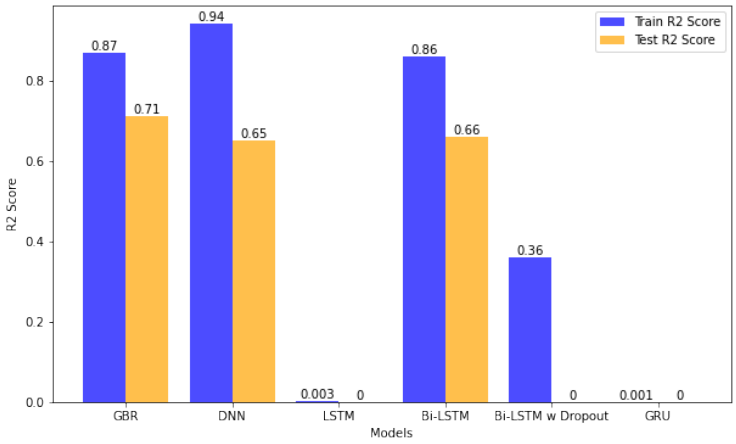


Fig. A.1: R^2 - No Differencing

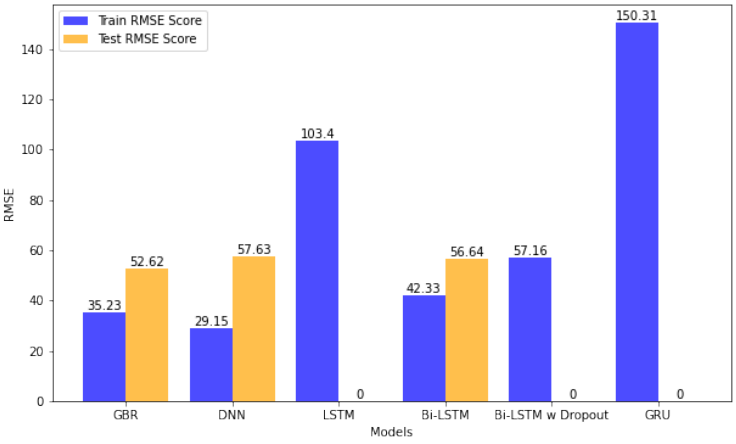


Fig. A.2: RMSE - No Differencing

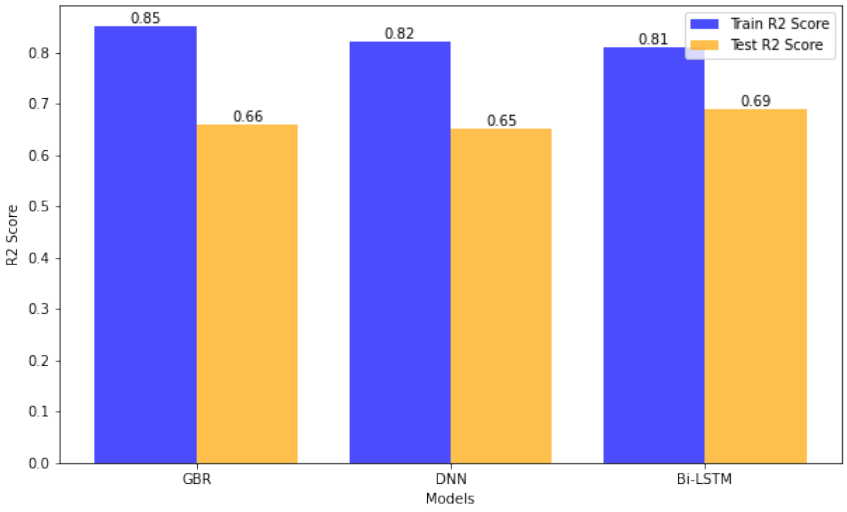


Fig. A.3: R^2 - First Order Differencing

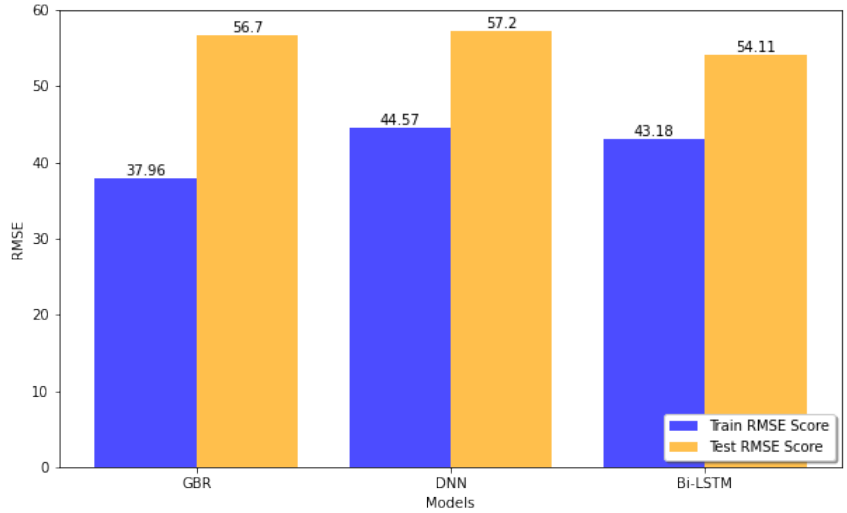


Fig. A.4: RMSE - First Order Differencing

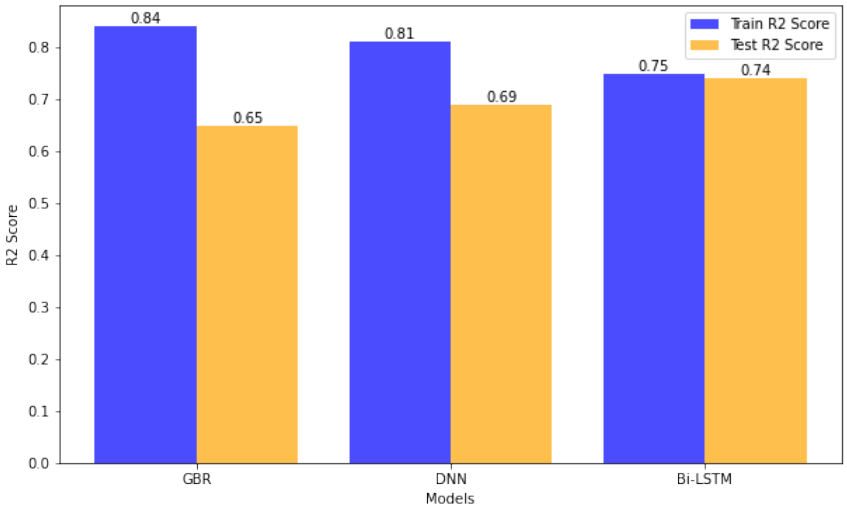


Fig. A.5: R^2 - Second Order Differencing

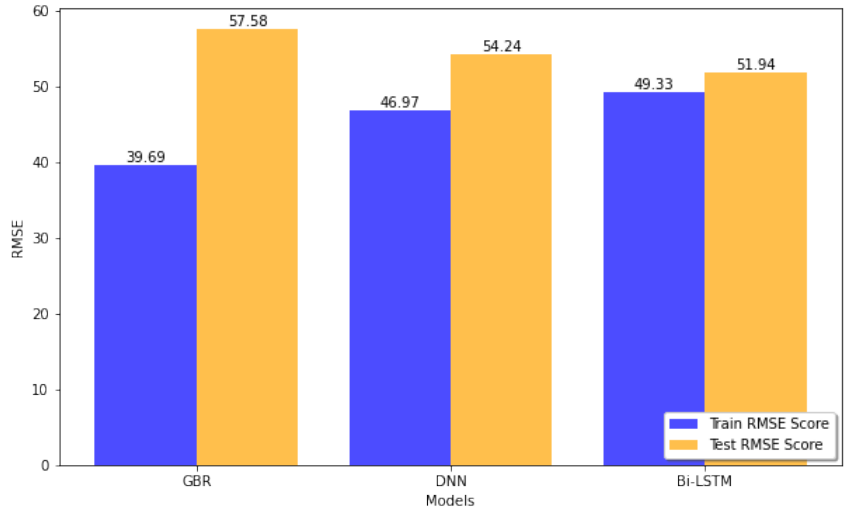


Fig. A.6: RMSE - Second Order Differencing

4.2. Interpolated Data

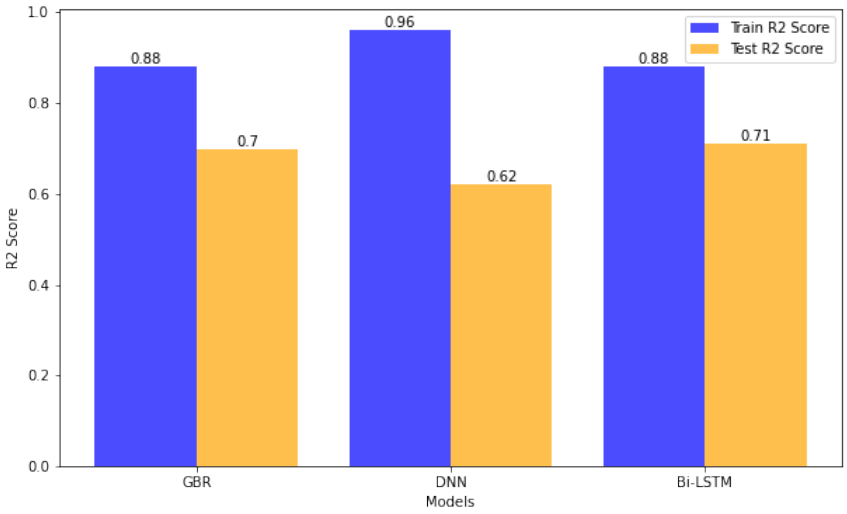


Fig. A.7: R^2 - No Differencing

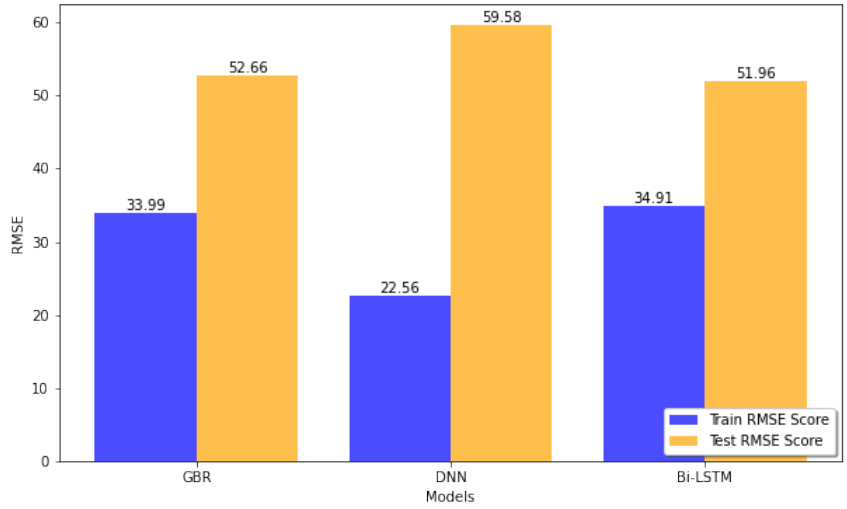


Fig. A.8: RMSE - No Differencing

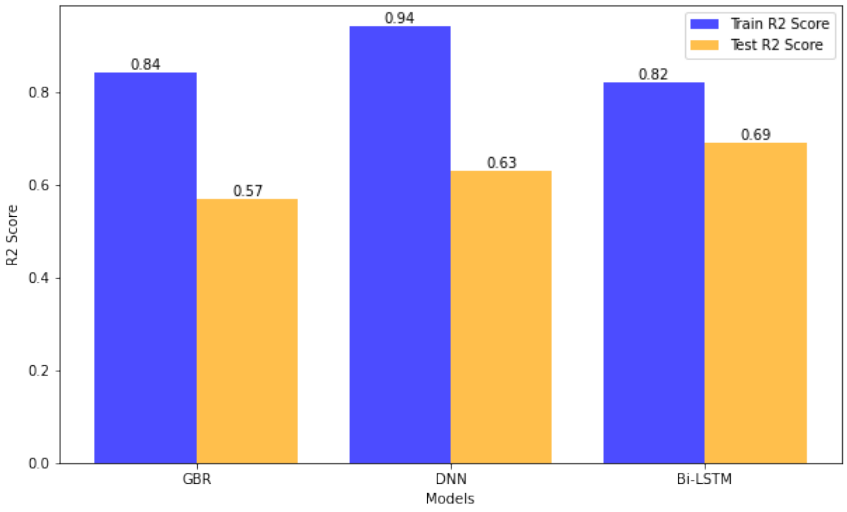


Fig. A.9: R^2 - First Order Differencing

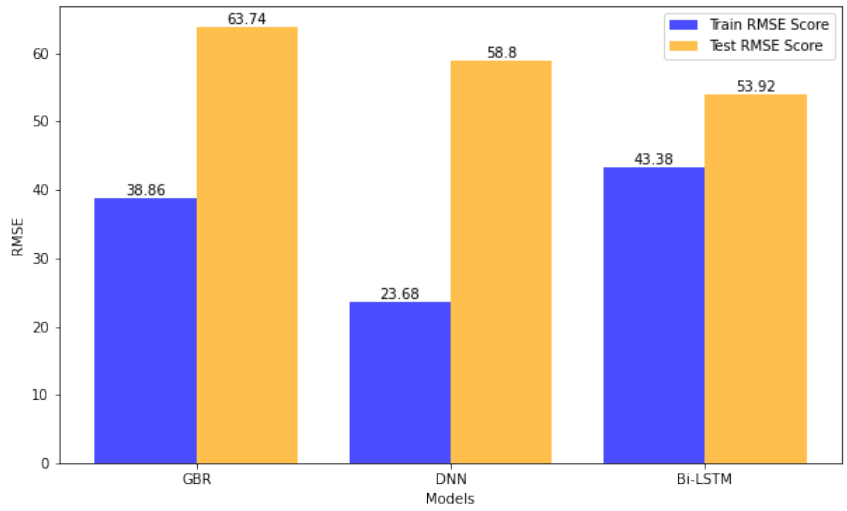


Fig. A.10: RMSE - First Order Differencing

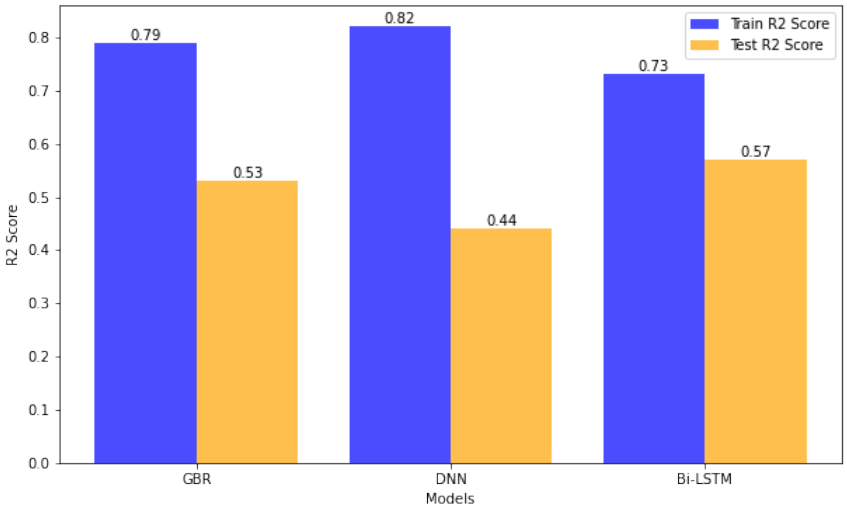


Fig. A.11: R^2 - Second Order Differencing

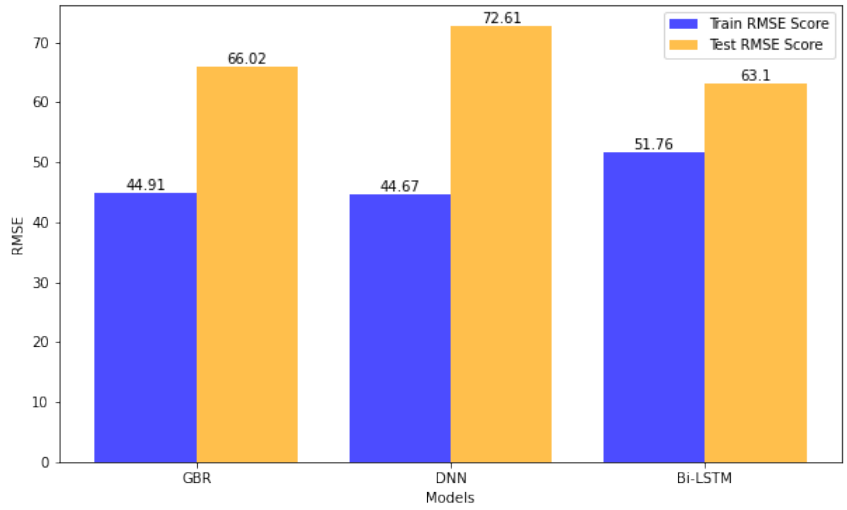
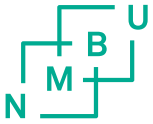


Fig. A.12: RMSE - Second Order Differencing



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway