Deep Learning Based Approaches for Imputation of Time Series Models

by

Muhammad Saad

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see *Statement of Contributions* included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Five publications have resulted from the work presented in the thesis:

1. L. Nassar, I. E. Okwuchi, M. Saad, F. Karray, K. Ponnambalam and P. Agrawal. "Prediction of strawberry yield and farm price utilizing deep learning". (Accepted to IEEE World Congress in Computational Intelligence 2020)

2. L. Nassar, I. E. Okwuchi, M. Saad, F. Karray and K. Ponnambalam. "Deep learning based approach for fresh produce market price prediction". (Accepted to IEEE World Congress in Computational Intelligence 2020).

3. M.Saad, M.Chaudhary, F.Karray and V. Gaudet. "Machine Learning Based Approaches for Imputation in Time Series Data and their Impact on Forecasting." (Accepted to IEEE International Conference on Systems, Man and Cybernetics 2020)

4. M.Saad, L.Nassar, F.Karray and V.Gaudet. "Tackling Imputation Across Time Series Models Using Deep Learning and Ensemble Learning." (Accepted to IEEE International Conference on Systems, Man and Cybernetics 2020)

5. L.Nassar, M.Saad, I. E. Okwuchi, M.Chaudhary, F.Karray and K. Ponnambalam. "Imputation Impact on Strawberry Yield and Farm Price Prediction Using Deep Learning." (Accepted to IEEE International Conference on Systems, Man and Cybernetics 2020)

In papers 1 and 2, I was responsible for data preprocessing and building the deep learning model which includes LSTM. The work in papers 1 and 2 formed part of sections 4.3.1 in the thesis.

For paper 3, I performed the comparison between the statistical, machine learning and deep learning models and also gauged its impact on prediction. The work in paper 3 formed part of sections 4.2.1 in the thesis.

In paper 4, I created the recommendation framework for imputation of various types of time series data. The content of this work form part of section 4.2.2 and 4.2.3 in the thesis.

In paper 5, I was responsible for data preprocessing, model building and determining the impact of imputation on forecasting. The content of this work appear in sections 4.3.1 of the thesis.

Abstract

Market price forecasting models for Fresh Produce (FP) are crucial to protect retailers and consumers from highly priced FP. However, utilizing the data for forecasting is obstructed by the occurrence of missing values. Therefore, it is imperative to develop models to determine the value for those missing instances thereby enabling effective forecasting. Usually this problem is tackled with conventional methods that introduce bias into the system which in turn results in unreliable forecasting results. Therefore, in this thesis, numerous imputation models are developed alongside a framework enabling the user to impute any time series data with the optimal models. This thesis also develops novel forecasting models which are used as a gauging mechanism for each tested imputation mode. However, those forecasting models can also be used as standalone models.

The growth and success of deep learning has largely been attributed to the availability of big data and high end computational power along with the theoretical advancement . In this thesis, multiple deep learning models are built for imputing the missing values and also for forecasting. The data used in building these deep learning models comprise California weather data, California strawberry yield, California strawberry farm-gate prices, USA corn yield data, Brent oil type daily prices and a synthetic time series dataset. For imputation, mean squared error is used as an metric to gauge the performance of imputation whereas for forecasting a new aggregated error measure (AGM) is proposed in this thesis which combines mean absolute error, mean squared error and R^2 which is the coefficient of determination.

Different models are found to be optimal for different time series. These models are illustrated in the recommendation framework developed in the thesis. Different stacking ensemble techniques such as voting regressor and stacking ML ensemble are then utilized to have better imputation results. The experiments show that the voting regressor yields the best imputation results. To gauge the robustness of the imputation framework, different time series are assessed. The imputed data is used for forecasting and the forecasting results are compared with market deep and non-deep learning models. The results show the best imputation models recommended based on work with the synthesized datasets are in fact the best for the tested incomplete real datasets with Mean Absolute Scaled Error (MASE) <1 i.e. better than the naive forecasting model. Also, it is found that the best imputation models have higher impact on reducing the forecasting errors compared to other deep or non-deep imputation models found in literature and market.

Acknowledgements

I thank my supervisors Professor Fakhri Karray and Professor Vincent Gaudet for all the guidance and support that they provided during my Master's degree. I also thank them for the opportunity to work on this research project. I also thank all my colleagues and team members of this project: Dr. Lobna Nassar, Professor Jamshid Mousavi and Mohita Chaudhary. I would also like to acknowledge their contributions: Dr. Nassar's contribution to experiment ideas, helping with data extraction, result compilation, documentation and revision and Ms. Chaudhary's help with data preprocessing.

I also wish to thank the members of my thesis committee, Professor Mark Crowley and Professor Kumaraswamy Ponnambalam for taking the time to provide an in-depth review of the thesis and provide very helpful feedback. Your efforts and support are deeply appreciated.

I acknowledge and am thankful for the financial support from Loblaw Companies Limited, Natural Sciences and Engineering Research Council, University of Waterloo and Electrical and Computer Engineering department.

I thank my parents Mr Shahid Muhammad Dean and Mrs Aisha Shahid for their constant love, prayers, support and sacrifices that allowed me to pursue my Master's degree. My deepest appreciation also goes to my siblings, relatives and friends for their support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGM	Aggregated Error Measure
AI	Artificial Intelligence
ANN	Artificial Neural Network
CIMIS	California Irrigation Management Information System
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short Term Memory
DBN	Deep Belief Networks
DC	Distribution Center
DL	Deep Learning
DNN	Deep Neural Network
ETo	Evapotranspiration
FC	Food Company
FP	Fresh Produce
GARCH	Generalized Autoregressive Conditional Heteroskedastic
GBR	Gradient Boosting Regressor
GRU	Gated Recurrent Unit
LR	Linear Regression
LSTM	Long Short Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error

ML	Machine Learning
PCA	Principal Component Analysis
R^2	Coefficient of Determination
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SES	Simple Exponential Smoothing
SVM	Support Vector Machine
SVR	Support Vector Regressions
VAR	Vector Autoregression
VR	Voting Regressor
WTI	West Texas Intermediate
XGBoost	Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Problem Definition

Well-established firms related to the food industry (FC) in Canada employ a large number of labours and also ensure to deliver fresh produce (FP) items in other parts of the country from their Distribution Centre (DC). This means that these firms play a crucial role in upholding the economy of the country by providing employment and fulfilling the needs of the people. Ensuring that all the demands are fulfilled, these firms take in large warehouses as their DCs at which certain temperature conditions are maintained in order to avoid deterioration of the food quality. These warehouses are also responsible for ordering and distributing the produce to various retailers.

The supply chain of a fresh produce (FPSC) is not as simple as it might sound, there are multiple parameters that play an important role in making it a complicated task. For instance, the ordering of FP is dependent on the fluctuating procurement prices which in turn is dependent on the expected crop yield and other factors. Hence proper timing in terms of ordering is really vital to minimize waste and maximize profits. For example, U.S. estimates an annual loss of 5.9 and 6.1 billion pounds for fresh fruit and fresh vegetables, respectively [17] and a big part of these losses can be attributed to poor planning. In short, prices are dependent on multiple factors related to supply (weather conditions, soil, irrigation, etc.) and demand (environment, culture, consumer elasticity, etc.). These factors may also be affected by high uncertainty due to environmental and socioeconomic effects such as income, labor and other trade issues. Due to the fast paced technological advancement, today many of these factors are becoming even more uncertain which makes the procurement task a challenging one.

1.2 Motivation

The FP procurement process follows a form of bidding process where the buyer (DC) makes an offer to the vendor or farmer and the decision to sell or not is up to them. This means that having the estimate for the future minimum cost that the farmer or vendor might accept is critical. This is because these procurement prices are important for the DC owning firms to survive in today's economy. Due to the great volume of transactions and monetary value (over $10 billion by the food company), a saving of even a micro cent in each FP transaction transfers to benefits of tens of millions of dollars each year for Canada. The resulting benefits would be a lower FP prices at the consumer level and less wastage or losses at the corporate level. Hence, it is imperative to develop a forecasting model to predict the prices thereby maximizing the profits. However, one problem which hinders this activity is the problem of missing values in the dataset. These missing values may be caused by various factors ranging from human mistakes to device or equipment noise/malfunctioning [149, 85]. In order to mitigate this issue various methods are employed and the accuracy of those methods plays an essential role in making the results of the forecasting model reliable and accurate. Therefore, this research tackles this problem by introducing various novel imputation frameworks and models to provide greater accuracy and gauge its impact on FP forecasting. Utilizing these models is a new research problem which has not yet been tackled by anyone.

The success of the imputation models is due to the availability of large computational resources and development of powerful AI algorithms. These factors aid in making reliable imputation models with high accuracy. By utilizing these models, the data fed to the AI forecasting models is wholesome, helping the models to make more reliable predictions which in turn aids the firms to have a greater profitability and consumers to have greater affordability.

1.3 Scope

This research focuses on the development of imputation models which range from statistical to complex deep learning models. These models were developed for multiple applications. Despite the fact that the framework used for imputation is utilized in various applications, limited application is seen in the domain of FP. For this research, strawberry is chosen as the case study and studies show that the majority of strawberries in Canada are procured from Southern California, so datasets covering that region are considered.

1.4 Objective

The main objective of this work is to impute various types of datasets using the most suitable imputation models in order to enhance the precision of any forecasting model using these datasets as input. To reach this main objective, the following steps are carried out:

1. Compare imputation models of different domains (statistical, machine learning and deep learning) on complete datasets to explore the best performing domain by developing complex models in that domain.
2. Collect data of the parameters that affects the strawberry produce which includes weather, yield and price data.
3. Develop forecasting models for forecasting yield and price.
4. Perform forecasting using the unimputed datasets then compare the results with those attained from forecasting using the datasets imputed with the developed models.
5. Evaluate the models by comparing them against market models to gauge the effectiveness of the built models.

1.5 Thesis Organization

This thesis comprises five chapters. The current chapter gives an overview of the problems encountered in FP procurement and how solving this unique problem can be beneficial to businesses and society. It went on to outline the scope and the objectives of this research. Chapter 2 covers the background and literature review. It reviews FP procurement, time series modelling, machine learning techniques, performance evaluation and related work. In chapter 3, the proposed solution is described including proposed models and our proposed evaluation metric. Chapter 4 provides details of the various experiments carried out. Finally, chapter 5 summarizes the major findings derived from this research work.

Chapter 2

Background and Literature

2.1 Fresh Produce (FP) Procurement

In today's fast paced economy, improved understanding of market signals, whether it be price or any other factor, is imperative to facilitate not only better decision making but also to point out opportunities to extract greater value for food companies. Fair pricing is a common economics term which refers to the situation where market equilibrium, supply and demand curves intersection, results in a price at which the participants, buyers and sellers, attain normal profit over time [2]. The work in this research looks into opportunities for added value that can be achieved at the FP procurement field using numerous artificial intelligence (AI) and machine learning (ML) algorithms.

To enable the utilization of the AI-based systems, it is important to prepare the data and make it acceptable for the algorithms. One of the problems that hinders this process is having missing values. Prior to AI techniques, various non-machine learning models were used to solve this issue, but with the recent advancement in AI, there is a great improvement in imputation which ultimately results in better decision making in terms of procurement. To place good procurement orders, it is vital to have a clear and complete view of the costs and value-adding factors that determine fresh produce prices over time. For instance, the supply side is affected by external factors such as climate change, government policies etc. Whereas, the demand side is affected by the changing preferences of consumers such as preference of purchasing organic food rather than inorganic.

The factors that influence the pricing of the fresh produce in the Canadian markets [128] are:

1. The relationship between the farm gate and retail prices
2. The stages through which the produce go

Studies show that the farmer's share of retail value is significant for the FP [2]. This means that the factors that impact production cost and yield are paramount for forecasting models and therefore having complete data for those factors is necessary. Retail price is dependent on, not only farm gate price, but also on transportation, distribution and storage costs. Additionally, there are 4 significant regional and seasonal price variations mainly due to supply and demand. These analyses are of paramount importance in the data pre-processing stage.

Although there are many supply regions in Canada, the majority, over 50%, of the fresh produce is imported from California, which is undergoing significant changes in many factors that affect FP. Some of the examples are the changing weather, falling groundwater levels and labor. Recently, there is another development in which farmers are demanding more share of water to mitigate their needs of food growing [118]. These factors greatly influence the price of FP and hence consideration of these parameters is imperative.

Since the business costs are commercially confidential, market knowledge and intelligence are the key features to fill the gap and help the procurement professionals to negotiate their orders. Industry efforts to improve transparency are highly dependent on the capabilities of the organizations and their willingness to share data. The FC team provides us their data, information and experiences, e.g. past procurement transactions, that are inaccessible otherwise. Publicly available climate-related data and agricultural data are also collected to train, calibrate and validate the AI models.

2.2 Time Series (TS) Modeling

A time series is a set of observations x_t recorded over the period of time t [16]. Amongst numerous applications of time series modeling the most widely used application is time series forecasts in different sectors such as health care, financial market, etc. [79]. The two most commonly used approaches for time series forecasting are exponential smoothing and the auto-regressive integrated moving average (ARIMA) models. While exponential smoothing models are based on the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data [60]. The main difference of utilizing time series methods as opposed to other methods is that time series methods take into account the internal structure of the data that plays an important role in the forecasting [16].

2.2.1 Univariate Time Series

Univariate time series refers to the time series that has scalar observations over the period of time. A classical example would be the daily maximum temperature in Waterloo, Canada. Univariate methods include time series forecasting methods [59, 66, 30], which use previous prices to predict the future price. Univariate TS may possess the following characteristics:

1. Trend: A trend time series moves in a simple linear fashion. The trend shows the general tendency of the data to increase or decrease during a long period of time. The increase or decrease does not need to be in the same direction throughout the given period of time [153].

2. Seasonality: A seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week). Seasonality is always of a fixed and known period. Hence, seasonal time series are sometimes called periodic time series [50].

3. Combined: Many time series data contain the trend and seasonal patterns as its basic components. Trend and seasonality are very commonly encountered in economic and business time series [43].

4. Stationarity: It is an important characteristic of time series. A time series is referred as stationary if its statistical properties do not change over time. In other words, it has a constant mean and variance and it is independent of time [71]. Stochastic time series can be classified as stationary, e.g. the autoregressive (AR), moving average (MA), and auto-regressive moving average (ARMA) [77], and the non-stationary ones, e.g. the auto-regressive integrated moving average (ARIMA) [45] and the generalized autoregressive conditional heteroskedastic (GARCH) [46]. All these models are used for short time horizons.

2.2.2 Multivariate Time Series

Contrary to univariate time series, multivariate time series has more than one time dependent variable. This means that each variable not only depends on its own past values but it also depends on the other variable values [19]. This cross dependency among variables influences the forecasting and hence it is important to incorporate that in the models. In terms of multivariate time series, x_t includes multiple univariate time series which contribute towards forecasting y_{t+1}. The choice of these series is guided via both empirical and domain knowledge [119].

Due to cross dependencies of the variable, multivariate time series models have large number of unknown parameters and these parameters increase drastically when non-linearity is introduced. Ideally the concept of extending the univariate time series models to multivariate seems straightforward but in practice, it is difficult to conclude on what the best approach is because some level of experimentation has to be done to determine the best approach [119].

2.3 Nondeep Machine Learning Models

The volatility in agricultural commodities makes the simple models less reliable and the complex ones less robust. In order to overcome this shortcoming, there is a need to use ML models that not only can cater the non-linearities in the data but are also robust enough to be generalized. Today with the advent of big data and advancement in computation power, utilization of ML models is found to be convenient. The ML model takes into account all factors that impact cost and yield of the FP and that is why it is also considered as multivariate model.

ML models can be broadly classified into two main categories which are: supervised and unsupervised learning. Supervised learning is the technique in which the model is fed the input along with its output and the model is expected to find the relationship between the two. On the other hand, the unsupervised learning is a technique in which the ML models find the patterns within the data itself. The main difference between the two is that in the supervised learning the output is provided to the model whereas this is not applicable in the unsupervised learning. A large percentage of machine learning applications today are built using supervised learning. Supervised learning is applied to classification and regression problems.

Forecasting and imputation both fall in the domain of supervised learning, specifically they utilize the regression algorithm. ML algorithms within this sub-group include deep learning as well as non-deep learning models. The linear regression (LR), support vector regression (SVR), K-nearest neighbours and gradient boosting regression (GBR) are examples of the nondeep learning ML models. More details regarding these algorithms are outlined below.

2.3.1 Linear Regression

Linear regression (LR) is the most basic approach for determining the relationship between the dependent and independent variables. In case there is a single independent variable, it is known as simple linear regression. On the contrary, having multiple independent variables changes the name of the model to multiple linear regression [38].

The relationships in the linear regression are determined using the linear predictor functions whose model parameters are estimated from the data. These models are referred to as linear models [111]. Similar to the other regression analysis, linear regression looks into the conditional probability distribution of the dependent variable given the values of independent variables [102].

Consider an example where $X_i = \{x_1, x_2, ..., x_n\}$ is the vector of independent variables while y_i is the dependent variable. The linear regression model aims to find the relationship between the two and in doing so it finds a plane that can illustrate this relation. The relationship is expressed as follows:

$$y_i = \beta_o + \beta_1 x_i + \beta_2 x_2 + ... + \beta_n x_n \tag{2.1}$$

By vectorization we can reduce the eq. (2.1) as:

$$y_i = X_i \beta^T \tag{2.2}$$

where β is the row vector of $\beta_o, \beta_1, \beta_2, .., \beta_n$ and X_i is the row vector of all the independent variables , $x_1, x_2, x_3, .., x_n$. The product $X_i \beta^T$ indicates the dot product between the two vectors.

2.3.2 Support Vector Regression (SVR)

The Support Vector Machine (SVM) is an elegant machine learning algorithm proposed by Cortes and Vapnik [29]. The basic concept behind the algorithm is structural risk minimization. It is a concept in which the generalization is achieved by balancing the model complexity against its ability of fitting the training data. The regression version of SVM is proposed in [36] and is widely used in time series prediction and imputation [124, 10].

Consider a time series where $X_i = \{x_1, x_2, x_3, ..., x_n\}$ is the vector of independent variables while y_i is the output and N is the total number of observations. The regression function for such a time series will be defined as:

$$y_i = f(X_i) = W^T \phi(X_i) + b \tag{2.3}$$

where W is the weight vector, b is the bias, and $\phi(X_i)$ is the function that maps the input vector to the higher dimension feature space. W and b are obtained using the following optimization function:

$$\begin{aligned}
\text{min} \quad & \frac{1}{2}\|W\|_2 + C\sum_{i=1}^{N}(\epsilon_i + \epsilon_i^*) && \text{(2.4a)}\\
\text{subject to} \quad & y_i - W_T(\phi(x)) - b \leqslant \epsilon + \epsilon_i && \text{(2.4b)}\\
& W^T(\phi(x)) + b - y_i \leqslant \epsilon + \epsilon_i && \text{(2.4c)}\\
& \epsilon_i + \epsilon_i^* \geqslant 0 && \text{(2.4d)}\\
& && \text{(2.4e)}
\end{aligned}$$

where C is the predefined regularization parameter which maintains the balance between the model simplicity and generality, ϵ_i and ϵ_i^* are the cost of the errors. SVR has the capability to cater for non-linearities by using a kernel function and that is the main reason SVR is preferred in time series.

2.3.3 K-Nearest Neighbours (KNN)

The k-nearest neighbours algorithm (KNN) is a non-parametric method proposed by Thomas Cover used for classification and regression [7]. In regression as well as in classification, the input consists of the k closest training examples in the feature space. The output depends on whether KNN is used for classification or regression:

- In classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In KNN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

KNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation. Since this

algorithm relies on distance for classification, normalizing the training data can improve its accuracy dramatically [92, 56]

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. A common weighting scheme involves giving each neighbor a weight of $\frac{1}{d}$, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (classification) or the object property value (regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

2.3.4 Decision Tree Based Models

Decision trees are a prominent method in machine learning [105]. The main reason for their popularity is the human understandability that they pertain unlike other algorithms. The main objective of the model is to predict the value of the target given a set of input variables.

In case of classification, all input features are considered to have finite and discrete domains and there is a single target label against each sample. The group of members sharing the same target label is referred as a class. A decision tree is a tree in which the input variables are represented by the nodes and the arcs from each node signify either the possible value of the target class or they lead to other input variables. Every tree leaf is labelled with a class or probability distribution over the classes, meaning that the tree has categorized the data set into either a specific class or a particular probability distribution [113, 98]. In simpler terms, a decision tree model is a tree-like model which learns the pattern in the data via if-else decision rules. The complexity of the model is proportional to the depth of the tree.

The concept of a decision tree can also be extended to regression problems [15]. The regression tree also works in a similar fashion except that in a regression tree, the value in the terminal node is the mean of the observations falling in that region [103]. Hence in case an unseen observation falls in that region, mean value is taken as the prediction result as opposed to classification where the class label is obtained at the terminal node. A common feature that both tree types have is dividing the independent variables into non-overlapping regions and following a top-down greedy approach known as binary splitting [98]. One downside of decision trees is that it is prone towards overfitting which ultimately leads to poor accuracy on unseen data. In order to tackle overfitting, pruning is used.

Pruning is a technique which reduces the number of leaves and nodes thereby reducing the model complexity.

The top-down approach followed by that decision tree is accomplished by selecting the variables that result in the best split for the input [103]; different algorithms are deployed for this purpose. The algorithms generally measure how homogeneous the target variable is within the subsets. Some examples of these metrics are residual sum of squares, Gini impurity, information gain and variance reduction [15].

Another way to mitigate the problem of overfitting is using the combination of multiple decision trees. The most common combinations are boosted trees such as gradient boosting and bagging trees such as random forests [39]. Gradient boosted trees are perceived as a machine learning technique applicable to both classification and regression. The model begins by fitting an initial model to the data then fitting another model in order to overcome the shortcomings of the initial model [40]. The process is continued until the overall prediction error is minimized. Combining the best possible model with the previously found models minimizes the overall prediction error.

The decision tree models of fixed size are used as weak learners on different subsets of the data. The predictions from all the weak learners are then combined to make a final prediction. The gradient boosted decision trees, just like the boosting algorithm, build an additive model in a greedy way that can be represented by Equation (2.5).

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{2.5}$$

where $h_m(x)$ is the newly added tree that tries to minimise the loss function L of previous model which is F_{m-1} in boosting. It is also known as the weak learner.

$$h_m = \arg\min_h \sum_i^n L(y_i, F_{m-1}(x_i) + h(x_i)) \tag{2.6}$$

The steepest descent method is used to minimize the error, in this method the direction of the steepest descent is the negative gradient of the loss function evaluated at the current model which can be calculated for any differentiable loss function.

$$F_m = F_{m-1}(x) - \gamma_m \sum_{i=1}^n \nabla_F L(y_i, F_{m-1}(x_i)) \tag{2.7}$$

Here, γ_m is the size of the step and is given by:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^n L(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)} \tag{2.8}$$

2.4 Deep Machine Learning Models

Deep Learning (DL) is a subfield of machine learning inspired by the structure and function of the brain. Despite the wonders of nondeep machine learning techniques, they are limited in terms of processing data in its raw form. This means that careful engineering and considerable domain expertise are required to design a feature extractor that can convert the raw data into a form suitable for the nondeep machine learning models [52]. DL methods are representation learning methods that possess multiple levels of representation. This is attained by combining simple non-linear modules that transform the representation from a raw input level to a higher and slightly more abstract level. Using this composition, DL methods are able to learn complex functions [52].

Due to the robustness and increased capabilities, a lot of research is performed in exploring multiple architectures of DL. Some examples are deep neural networks (DNN), deep belief networks (DBN), recurrent neural networks (RNN), convolutional neural networks (CNN) etc. The foundation of these architectures is the artificial neural network (ANN). These architectures are applied to different fields including, computer vision, natural language processing, machine translation, speech recognition, etc., where they have produced results comparable to and in some cases better than human expert performance [26, 69].

2.4.1 Artificial Neural Network (ANN)

The artificial neural networks represent an important tool in DL. As the name suggests, this technique mimics the activity of a human brain. The human brain is perhaps the most complex living structure on the face of the earth. As per current knowledge, the brain is composed of about 86 million neurons which have over 100 trillion interconnections amongst themselves. Artificial neural networks are also composed of neurons and they replicate the structure of neural activity of a human mind. These systems 'learn' to carry out tasks by going through examples, usually without needing task-specific rules; supervised learning.

A lot of improvement has been made in ANNs to solve a large variety of problems such as computer vision, speech recognition, machine translation, playing board and video games, medical diagnosis, forecasting, anomaly detection, etc. [110]. An ANN is composed of the following:

- Neurons: These are the units that receive input, combine it with the internal state and apply a non-linear activation function to it to produce output. Activation functions have to be smooth while providing differentiable transitions with changes in input values.

- Connections and weights: Connections link the neurons which help input to propagate to become output. Each connection is assigned a weight which increases or decreases the significance of that connection.

As a machine learning algorithm, ANN learn to minimize the cost function. This is achieved by adjusting the connection weights to compensate for each error found during learning. This adjustment is attained via backpropagation (backprop) algorithm. Technically, backprop calculates the derivative of the cost function associated with a given state with respect to the weights. The weight is updated using stochastic gradient descent.

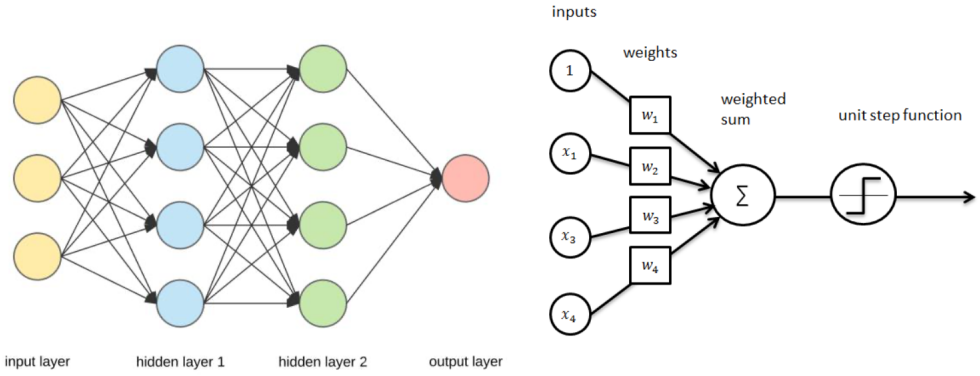

Figure 2.1. Artificial Neural Network (ANN) Architecture

2.4.2 Recurrent Neural Network (RNN)

Recurrent neural networks (RNN) are a class of ANNs where connections between nodes create a directed graph along a temporal sequence. RNNs are similar to traditional time series models as they are both able to model time dependent relationships in the data. In RNNs each node in a given layer is connected with a directed, one-way, connection to every other node in the next successive layer. Every node has a time-varying real-valued activation function and each connection (synapse) has a modifiable real-valued weight. Nodes are either: input nodes receiving data from outside the network, output nodes yielding results, or hidden nodes that modify the data enroute from input to output.

At time t, nodes with recurrent edges receive input from the current data point x_t and also from the hidden node value h_{t-1} in the network's previous state. The output $\hat{y}_t$ is

calculated and the output is transmitted to hidden node h_t at time t. This means that the input x_{t-1} at time $t-1$ influences the output $\hat{y}_t$ at time t and later by way of recurrent connections [74]. The computations carried out in the process is as follows:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{2.9}$$

$$\hat{y}_t = softmax(W_{yh}h_t + b_y) \tag{2.10}$$

where σ is the sigmoid activation function which is a non-linear transformation that maps the value between 0 and 1, W_{hx} is the matrix of conventional weights between the input and the hidden layer and W_{hh} is the matrix of conventional weights between the hidden layer and itself at adjacent time step. The vectors b_h and b_y are bias parameters which help neural networks to learn the offset [74]. The output $\hat{y}_t$ is the predicted value of the sequence.

The architecture of a neural network can be illustrated in Fig. 2.2. As evident from the figure, RNN can be interpreted not just as cyclic but also as a deep layer per time step with shared weights across the various time steps. The unfolded network can be trained across multiple time steps using backpropagation. Since in RNN we propagate through time, the backprop algorithm is named as backpropagation through time (BPTT) [136]. Despite the wonders of RNN, it suffers from couple of limitations. While training the network, when gradients are propagating back in time, all the way up to the initial layer, the gradients go through multiple simultaneous matrix multiplications and as a result of using Chain Rule, if they have values less than 1 (<1), they diminish exponentially until they become negligible or 'vanish'. This deems the network model impossible to learn anything as weights won't be updated. This is known as the 'Vanishing Gradient Problem.' Similarly, if the values of gradients propagating back in time are greater than 1 (>1), their values escalate and eventually destroy the model's capability to learn anything making it unstable. This is known as the 'Exploding Gradient Problem.'

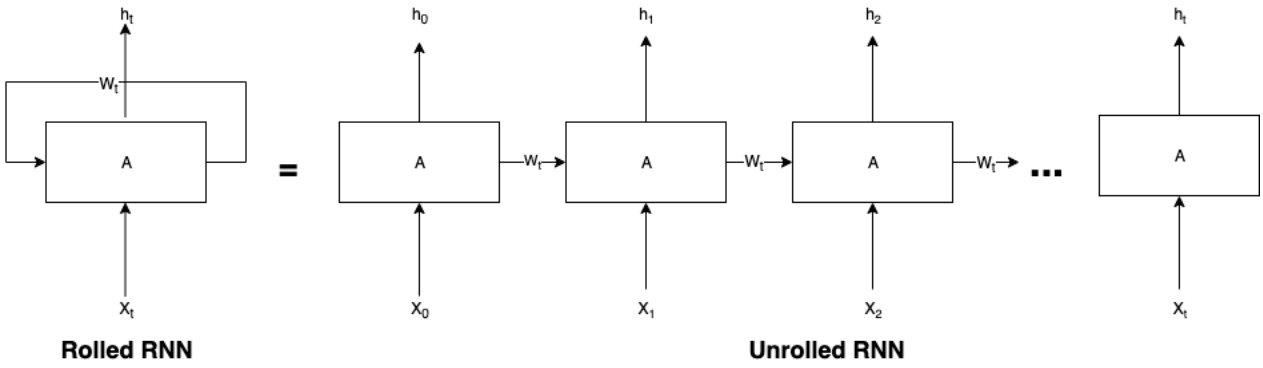

Figure 2.2. Recurrent Neural Network (RNN) Architecture

2.4.3 Long Short Term Memory (LSTM)

If a long sequence of data is fed to RNNs, they encounter a problem of carrying the information from the earlier time steps to a later one causing them to miss important information. Also, RNNs suffer through the problem of vanishing gradients. To overcome these issues LSTMs [57] are developed. They are called LSTMs as they can keep the short term memory for a longer time. The LSTMs are similar to the RNNs except that they have memory blocks in them. The memory blocks have gates through which the information is let through. The gates use sigmoid activation units to control whether they are triggered or not. The sigmoid function assigns a value between zero and one, which decides how much information should be let through the gates.

LSTMs have three gates: the forget gate, input gate, and output gate. The forget gate layer, which is a sigmoid layer, helps decide what information needs to be discarded. It looks at the previous hidden state h_{t-1} and the current input x_t and outputs a number between 0 and 1 for the cell state c_{t-1}, which decides how much information needs to be forgotten. It is denoted as f_t and is given as:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.11}$$

The input gate conditionally decides to update the memory state on the basis of the input. The input layer gate, which is composed of a sigmoid activation function, helps decide what information has to be updated. The next tanh layer in the input gate creates a vector of the new candidate values, which can be added to the state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i \tag{2.12}$$

$$\vec{C_t} = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.13}$$

In the next update layer the new vector $\vec{C_t}$ is combined with the previous cell state C_{t-1}. The old state is multiplied by f_t to forget the values that were decided to be forgotten initially.

$$C_t = f_t * C_{t-1} + i_t * \vec{C_t} \tag{2.14}$$

Then comes the output gate which decides what to output.

$$\sigma_t = \sigma(W_o[h_{t-1}, x] + b_o) \tag{2.15}$$

$$h_t = \sigma_t * \tanh(C_t) \tag{2.16}$$

The sigmoid decides which part of the cell state is generated. The output of the sigmoid is filtered through tanh. LSTMs are illustrated in the Fig. 2.3

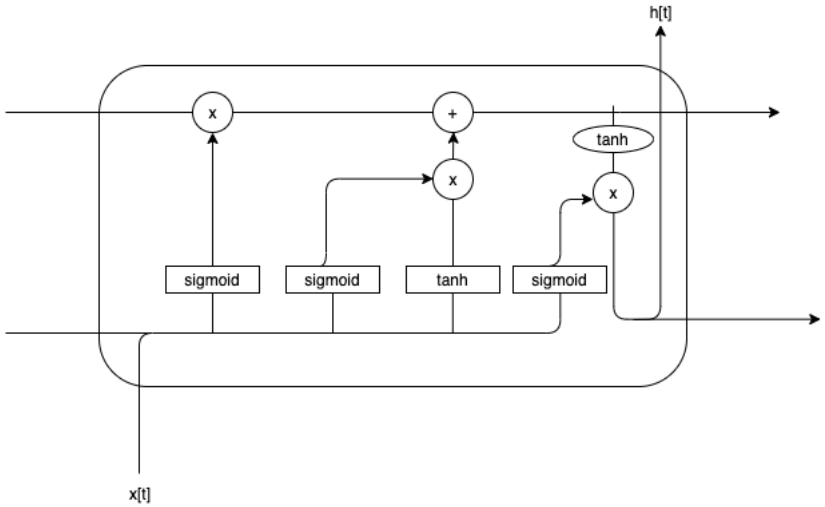

Figure 2.3. Illustration of the LSTM structure.

2.4.4 Gated Recurrent Unit (GRU)

GRU [33] is a newer version of RNNs and is quite similar to LSTMs [41]. GRUs use hidden states rather than using the cell state or memory to transfer the information. They have two gates, a reset gate and an update gate. The update gate acts similarly to the forget and input gates of LSTM. The reset gate, on the contrary, is used to decide how

much past information has to be forgotten. GRUs are faster than LSTMs since they have fewer tensor operations. Both LSTMs and GRUs are designed to overcome the short term memory issues faced by RNNs. Fig. 2.5 represents the GRU structure.

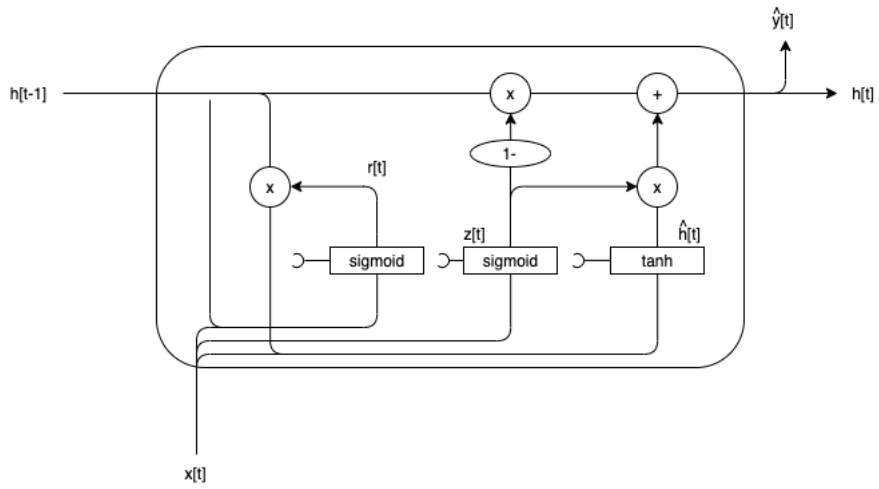

Figure 2.4. Illustration of the GRU structure.

2.4.5 Convolutional Neural Networks (CNN)

Convolutional Neural Networks also known as CNN or ConvNet are usually applied to image recognition and analysis. CNNs are also called shift invariant or space invariant artificial neural network since they use shared weights and translation invariance. Invariance to translation means that if we translate the inputs the CNN will still be able to detect the class to which the input belongs. In terms of time series, it implies that the pattern in the sequences is deciphered regardless of what part of the sequence these patterns appear [93]. CNNs have been applied to a variety of domains such as image analysis and detection [134, 3], recommender systems [130], natural language processing [28], time series modelling [127], etc.

One of the key features of CNN is that it is a regularized version of fully connected network. CNN tackles the problem of overfitting by employing weight sharing. For instance, in a fully connected network a small greyscale image of size 100 x 100 will cause each neuron to have weights of 10,000. CNN eradicates this issue by reducing the number of free parameters allowing the network to be deeper with fewer parameters [3]. The term

"convolutional neural network" comes from the implementation of a mathematical operation called convolution. Hence, CNNs are ANNs that use convolution in place of general matrix multiplication in their layers [52]. CNNs consist of an input and an output layer, as well as several hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a dot product. The most common activation function is RELU, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers. The various parts of a CNN are described as follows:

- Convolution: It is the building block of CNNs. A small matrix of number (also known as filter or kernel) is passed over the data and transform it based on its values. The kernel shifts from left to right and passes onto the data. This shifting is known as stride. In a CNN, the input is typically a tensor with size (number of images x image width x image height x image depth), so after passing the image through the convolutional layer, the resultant size becomes (number of images, feature map width, feature map height, feature map channels). Hence, in the network a convolutional layer should have a kernel defined by height and width, number of input and output channels and number of kernels. Building onto that, for time series problem a kernel is passed over 1D sequence using a sliding window. The primary purpose of sliding window is to convert the sequence to a supervised learning problem. The input sequence shape is modified to yield a tensor (number of sequences, sequence length which is the length of the 1D sequence, sequence height which takes the value 1, sequence depth which takes the value 1) so after convolution the shape becomes (number of sequences, feature map length, feature map height of 1, feature map channels of 1). This is essentially a 1D convolution. Unlike ANN, the same convolution filter, weights, is used across all time steps making the CNN learn filters that are invariant across time dimension.

- Pooling: : In order to reduce the dimensionality, pooling layers are utilized in CNN. Pooling layers reduce the dimension by combining multiple output of neurons and clustering them into one. There are two types of pooling layers which are, local pooling layer and global pooling layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer [27, 69]. In terms of operation, the most common operations performed is taking the maximum or average. Max Pooling takes the maximum value in the cluster and takes that as a single output [145, 26] while Average Pooling takes the average of the cluster and generates the output [84]. In this research for the time series problem 1D pooling is utilized. The loss of information during convolution and pooling does not reduce the

predictive power but rather helps extract the relevant information from the raw data into a smaller dimensionality which is much easier to learn from.

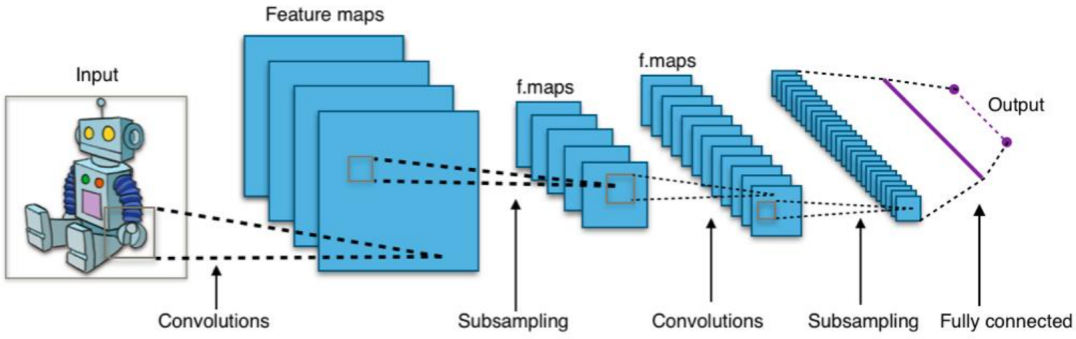

Figure 2.5. Typical CNN Architecture [9]

2.4.6 Convolutional LSTM (ConvLSTM)

LSTMs are extraordinary at recognizing temporal relationships yet they don't perform well in perceiving spatial connections. CNNs on the other hand are great in finding spatial connections however not temporal relations. To handle this, Convolutional LSTMs (ConvLSTM) are built which caters spatiotemporal connections simultaneously in the data [115]. In time series, spatial relationship refers to the pattern that exists based on the location of one data point relative to others. Whereas temporal relationship is the sequential order of the data points.

In comparison to conventional LSTM, ConvLSTM is able to cater the spatiotemporal structures by vectorizing the spatial information thereby overcoming the limitation of vector-variate representations in LSTM where spatial information is lost [135]. In ConvLSTM all the inputs $x_1, x_2, x_3, ..., x_t$, cell outputs $c_1, c_2, c_3, ..., c_t$, hidden states $h_1, h_2, h_3, .., h_t$ and gates i_t, f_t, g_t, o_t are 3D tensors in $\mathbb{R}^{PxMxN}$, where the first dimension is the number of samples, whereas the last two dimensions are the spatial dimensions. In order to utilize the model for time series forecasting, a slight modification is done. The first dimension is considered as the number of samples and the second dimension is taken as the sequence length whereas, the third dimension is set to 1. To get a better picture of the inputs and states, we may imagine them as vectors standing on a spatial grid. ConvLSTM determines the future state of a certain cell in the MN grid by the inputs and past states of its local

neighbors. This is easily achievable by using the convolutional operators in state-to-state and input-to-state transition [135].

Mathematically, it can be expressed as follows:

$$g_t = \tanh(W_{xg} * x_t + W_{hg} * h_{t-1} + b_g \tag{2.17}$$

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \odot c_{t-1} + b_i \tag{2.18}$$

$$f_t = \sigma(W_{xf} * x_t + w_{hf} * h_{t-1} + W_{cf} \odot c_{t-1} + b_f \tag{2.19}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.20}$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \odot c_t + bo \tag{2.21}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.22}$$

where σ is the element wise sigmoid activation funtion, * is the convolution operator and $\odot$ represents Hadamard product. If we view the states as the hidden representations of moving objects, a ConvLSTM with a larger transitional kernel should be able to capture faster motions while one with a smaller kernel can capture slower motions [115]. Similar to conventional LSTM, the gates in ConvLSTM help prevent gradient from vanishing by trapping it in the memory. The sigmoid and tanh activation functions here also work individually on all the components of the vectors. ConvLSTMs have been applied to precipitation nowcasting [115], air quality (PM 2.5) prediction [144], temperature prediction [73], video compression artifact reduction [143], etc.

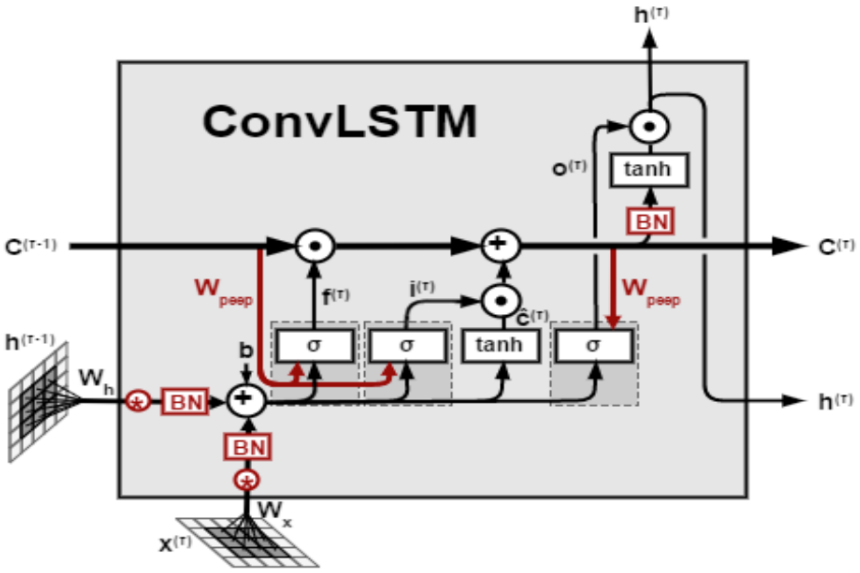

Figure 2.6. A ConvLSTM Cell [141]

2.4.7 SeriesNet

Since the traditional time series forecasting models cannot effectively extract good enough sequence of data features, researchers came up with a novel forecasting architecture named SeriesNet [114]. The SeriesNet consists of two networks. Typically, it is composed of an LSTM network and a dilated causal convolution network.

The dilated convolution is proposed to handle the loss of resolution or coverage due to the down-sampling operation in image semantic segmentation [150]. It uses dilated convolutions to systematically aggregate multi-scale contextual information and improve the accuracy of image recognition. The causal convolution is to ensure that the convolution kernel of CNN can perform convolution operations exactly in time sequence [131], and that the convolution kernel can only read the current information and historical information. The LSTM network, on the other hand, aims to learn holistic features and to reduce dimensionality of multi-conditional data.

The combined results of the networks help the models to learn multi-range and multi-level features from time series data, hence it has higher predictive accuracy compared to other models using fixed time intervals. Moreover, this model adopts residual learning

and batch normalization to improve generalization. The architecture of the network is illustrated in Fig. 2.7.

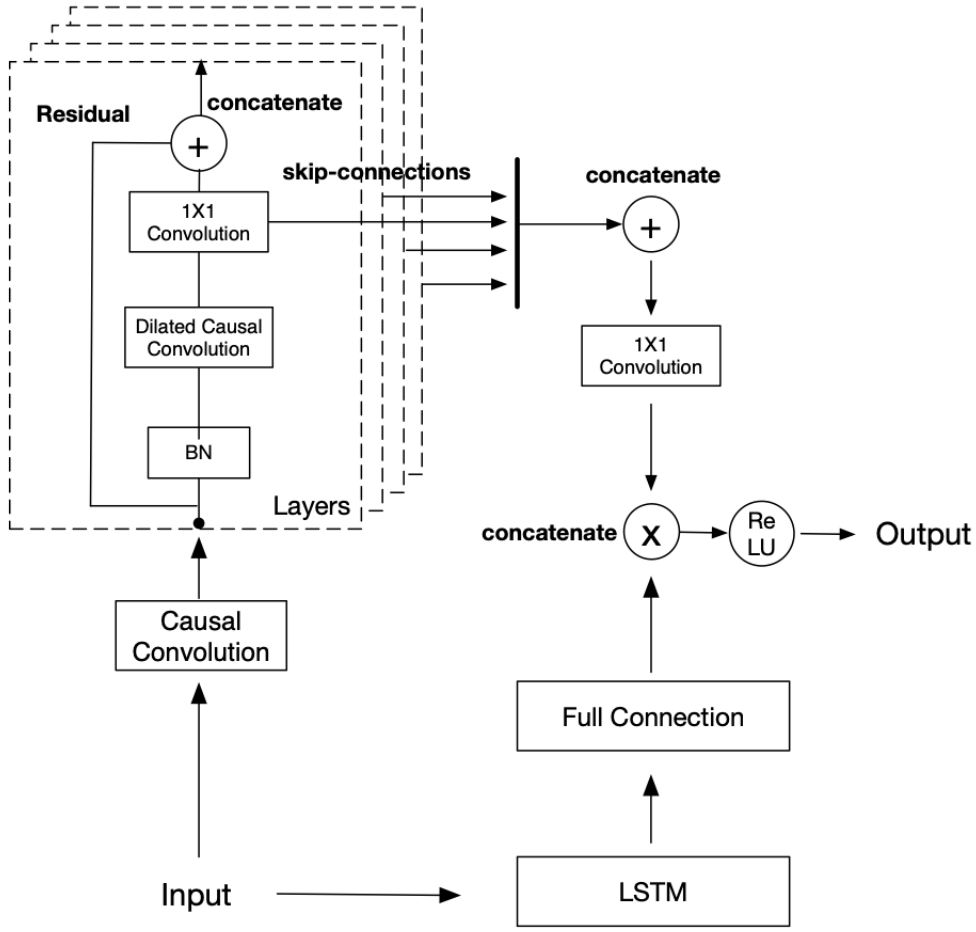

Figure 2.7. Architecture of SeriesNet [114]

2.4.8 Attention Mechanism

The attention mechanism is established to help memorize long source sentences in neural machine translation (NMT) [11]. Rather than building a single context vector out of the encoder's last hidden state, using attention allows creating shortcuts between the context

vector and the entire source input. The weights of these shortcut connections are customizable for each output element. In short, the purpose behind the attention mechanism is to aid the model to focus on the important aspects of the input data rather than considering all available information. An attention function is a mapping of a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors [132].

Self-attention mechanism which is also known as intra-attention mechanism is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading, abstractive summarization, or image description generation. There are different variants of attention; additive attention is the attention implemented in this work. This attention is computed as follows:

$$h_{t,t'} = \tanh({x_t}^T W_t + {x_{t'}}^T W_x + b_t) \tag{2.23}$$

$$e_{t,t'} = \sigma(W_a h_{t,t'} + b_a) \tag{2.24}$$

$$a_t = softmax(e_t) \tag{2.25}$$

$$l_t = \sum_{t'} a_{t,t'} x_{t'} \tag{2.26}$$

where σ is the element wise sigmoid function, W_t and W_x are weight matrices corresponding to ${x_t}^T$ and ${x'_t}^T$, W_a is the weight matrix corresponding to their non-linear combination and b_t and b_a are bias vectors [132]. Equation (2.26) explains how the attention l_t is calculated. In order to attain this, the probability distribution a_t in (2.25) of the compatibility score $e_{t,t'}$ in (2.24) is calculated first. This compatibility score is computed based on $h_{t,t'}$, the hidden representation ofx_t and$x_{t'}$ computed in (2.23).

2.5 Ensemble Learning

Ensemble pertains to combining results of two or more models and using them to perform the original task and this results in obtaining a better predictive performance than from the individual models [90, 94, 104]. Although machine learning models offer flexibility and

scalability to the amount of training data due to their non-linear behavior, a downside of this is that they learn via stochastic training algorithms, which makes them sensitive to the specifics of training data. This means that a different set of weights can be found each time the model is trained, which in turn produces different predictions. The resulting model is referred to as a model with high variance which can be strenuous when trying to develop one final prediction model. A successful approach for reducing the variance of the model is to train multiple models instead of a single one then combine their predictions [157]. This is called ensemble learning and it doesn't only reduce the variance of predictions but also results in better predictions. In short, ensemble learning is the process by which multiple models, such as classifiers, are strategically combined to solve a particular computational intelligence problem [95]. Common types of ensembles are outlined in the following subsections.

2.5.1 Bootstrap Aggregation (Bagging)

In this ensemble methodology, each model in the ensemble makes prediction and then the predictions from all the models are taken as votes with equal weights. In terms of regression, the average is taken of all the models' predictions. Furthermore, in order to minimize the variance, each model in bagging is trained on a randomly drawn subset of the training set. An example of bagging ensemble is Random Forest algorithm which combines multiple decision trees where each decision tree is trained on a randomly selected subset of the data [95, 13].

2.5.2 Voting Regressor

A voting ensemble works by combining the predictions from multiple models. It can be used for classification or regression. In the case of regression, this involves calculating the average of the predictions from the models. In the case of classification, the predictions for each label are summed and the label with the majority vote is predicted.

2.5.3 Stacking

A bagging ensemble or voting regressor combines the predictions from multiple trained models. A limitation of this approach is that each model contributes the same amount to the ensemble prediction, regardless of how well the model performs. A variation of this approach, called a weighted average ensemble, weighs the contribution of each ensemble

member by the expected performance of the model on a holdout dataset. This allows well-performing models to contribute more and low-performing models to contribute less. The weighted average ensemble provides an improvement over the model average ensemble. A further generalization of this approach is replacing the linear weighted sum model, e.g. linear regression, used to combine the predictions of the sub-models with any learning algorithm. This approach is called stacked generalization, or stacking for short. In stacking, an algorithm takes the outputs of the sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction. Stacking typically yields better performance than any single one of the trained models [138, 14].

2.6 Performance Measures

The metric that is chosen to evaluate the machine learning model is very crucial. The choice of metrics influences how the performance of machine learning algorithms is measured and compared. Most commonly used performance metrics in literature as in [4, 117, 61, 125, 96] [89, 90, 91] are the mean squared error (MSE), root mean square error (RMSE), mean absolute error (MAE), and mean absolute scaled error (MASE). Amongst them, R^2 is also a metric used to measure the degree of correlation between the predicted and actual values [5].

2.6.1 Mean Squared Error (MSE)

One of the most popular and preferred metrics is the mean squared error (MSE). It works by taking the average of the squared difference between actual and predicted values. Since it takes the square difference, it highly penalizes large errors. This metric is preferred mainly due to the fact that it is differentiable and hence it can be optimized. The only downside of MSE is that it is not robust to outliers [117, 61]. Mathematically, it is defined as follows:

$$MSE = \frac{1}{n}\sum(y_i - y_i')^2 \tag{2.27}$$

where y_i is the actual value whereas, y_i' is the predicted value and n is the sample size.

2.6.2 Root Mean Squared Error (RMSE)

RMSE is also widely used in regression problems. RMSE is the square root of MSE so it is a square root of the squared difference between the actual and predicted values. Since it takes in the squared difference, it penalizes large errors then the square root brings the error back to the same scale as of the target. This means that RMSE is advantageous when large errors are undesirable. It is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum(y_i - y_i')^2} \tag{2.28}$$

where y_i is the actual value whereas, y_i' is the predicted value and n is the sample size.

2.6.3 Mean Absolute Error (MAE)

Another metric is the mean absolute error (MAE). It calculates the absolute difference between the actual and predicted values, i.e. the errors, then calculates the average difference or error. MAE is sometimes preferred over MSE or RMSE since it is robust towards outliers. The only downside is that it is non- differentiable [137]. It is given as follows:

$$MAE = \frac{1}{n}\sum|y_i - y_i'| \tag{2.29}$$

where y_i is the actual value whereas, y_i' is the predicted value and n is the sample size.

2.6.4 Mean Absolute Scaled Error (MASE)

Mean absolute scaled error (MASE) is the mean absolute error of the forecast values, divided by the mean absolute error of the in-sample naive forecast. It was proposed in 2005 by Hyndman and Koehler, who described it as a "generally applicable measurement of forecast accuracy without the problems seen in the other measurements" [61]. MASE has a favorable property when compared to other methods for calculating forecast errors, such as RMSE, and is therefore recommended for determining accuracy of forecasts compared with naive forecasting [37]. The reason it is recommended as opposed to RMSE or other measure is because it is a scale-less measure and hence the scale of the sample does not impact it. The only downside is that it illustrated the accuracy of forecast in comparison to naive forecast. If the value is less than 1 (<1), the forecasting model is a good model

for forecasting whereas if the accuracy is greater than 1 (>1), the forecasting model is not a good model for forecasting. Mathematically it is defined as follows:

$$MASE = \frac{\frac{1}{J}\sum_j |e_j|}{\frac{1}{T-1}\sum_{t=2}^{T} |Y_t - Y_{t-1}|} \tag{2.30}$$

where the numerator e_j is the forecast error for a given period (with J, the number of forecasts), defined as the actual value (Y_j) minus the forecast value (F_j) for that period: $e_j = Y_j - F_j$, and the denominator is the MAE of the one-step "naive forecast method" on the training set which uses the actual value from the prior period as the forecast: $F_t = Y_{t-1}$

2.6.5 Coefficient of Determination (R^2)

The coefficient of determination is a statistical measurement that examines how differences in one variable can be explained by the difference in a second variable, when predicting the outcome of a given event. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model [18, 51, 35]. It measures the goodness of fit of a model and it lies between 0 and 1 [70]. However, in practical problems if the regression line is worse than using the mean value, the R^2 value will be negative. When a negative value occurs, it implies that the mean of the dataset fits the dependent variables better than the values provided by the model and there is a complete lack of fit [70]. This happens when an inappropriate model is chosen to solve a particular problem. TThe best result occurs when the predicted values are the same as actual values and that yield to R^2 being 1. Whereas, a model which always predict the mean will result in R^2 to be 0. In case where R^2 is 1, it implies that the residual sum of squares is low whereas, in case of $R^2 = 0$ the residual sum of squares will be high.

In this work and in other machine learning projects R^2 s normally used to provide a relative score of the model performance on scale of 0 to 1. Unlike other metrics like MAE or MSE, R^2 provides value which is more intuitive to gauge the model performance. Despite the wonders of R^2, it has some drawbacks; its value increases as the number of explanatory variable increases and it doesn't account for collinearity in the predictor variables. R^2 can be mathematically defined as follows:

$$SS_{Res} = \sum_n (y_i - y_i')^2 \tag{2.31}$$

$$SS_{Tot} = \sum_{n} (y_i - \bar{y_i})^2 \tag{2.32}$$

$$R^2 = 1 - \frac{SS_{Res}}{SS_{Tot}} \tag{2.33}$$

where SS_{Res} and SS_{Tot} are the residual sum of squares and total sum of squares respectively, y_i is the actual value, y_i' is the predicted value, $\bar{y_i}$ is the mean value and n is the sample size.

2.7 Related Work

Two aspects are tackled in this work: imputation and forecasting. The related work to each of these aspects is illustrated in the following subsections.

2.7.1 Imputation

Missing values imputation is an active research area and various methods exist for dealing with it. The most direct way to tackle this issue is to discard incomplete or empty records. Common examples are listwise deletion, which removes all instances with at least one missing value, or pairwise deletion, which removes an instance if the used variables contain missing values [58]. Unfortunately, these methodologies don't only introduce bias in the dataset but also reduce the amount of data available, thereby providing incorrect forecasting [156, 86, 109, 6]. Additionally, a method like linear interpolation is considered primitive. Despite the fact that this method fits a smooth curve to the given dataset and fills the missing values using local interpolations, this method fails to take into consideration the dependencies of features over time [42].

On the other hand, methods like SARIMA for seasonal [121] and ARIMA for non-seasonal time series [65] are auto-regressive methods yet their performance deteriorates when the dataset contains consecutive missing values. Nevertheless, better results might be attained by combining ARIMA and Kalman filter in the state space model [55].

Furthermore, imputation methods using K-nearest neighbors are also applicable for time series. In this approach, the method identifies k instances which are most similar to the missing value. Then a predefined rule, e.g. weighted average [126], or kernel function, e.g. exponential kernel [152], is applied to impute the values. Besides that, there are

certain pattern matching approaches as in [22], but these approaches work best only in certain cases.

Many researchers have also utilized Recurrent Neural Networks (RNN) to impute the missing values [21, 24, 75, 148]. Che et al. in [21] proposed a modified GRU, GRU-D, that imputes the missing values in a health care dataset. Its underlying assumption is that the missing variable can be represented as a combination of the last available value and the global mean. The GRU-D model has aided in harnessing the power of the RNNs [23] and the informativeness of the missing patterns in the data. The model uses GRU [25] and two representations of the informative missing patterns, which are masking and the time interval. Masking informs the model whether the input is present or not whereas the time interval illustrates the input observation patterns. The model is not only capable of capturing long term temporal dependencies of time series but also utilizing the missing patterns to improve the prediction results. GRU-D performed well on the health care dataset but it might exhibit limitations on other datasets.

Another state of the art algorithm that uses residual networks and graph-based temporal dependency in imputation is named LIME-RNN and is found to perform remarkable on various datasets [97]. In particular, it introduces a linear memory vector called the residual sum vector (RSV) and integrates it over previous hidden states of the RNN, to fill the missing values.

In this work, various novel AI based imputation models are built. Those models are not only compared with other DL domain model, such as LIME-LSTM, but also compared with the market models, such as mean imputation, median imputation, mode imputation, interpolation, Last observed carry forward, KNN-imputation, linear regression, support vector regressor and gradient boosting regression. Moreover, a recommendation framework is provided to decide the most optimal imputation strategy for each type of time series under consideration. Since the models use RNN and its variants, it is able to cater the long-term dependencies.

2.7.2 Time Series Forecasting

A lot of research is done in the domain of crop yield forecasting. The majority of this yield prediction has been for grains. Wheat yield prediction using ARIMA models was done in [100, 91, 83]. Rice time series forecasting has been tackled using statistical models as shown in [112, 101, 47]. More recently, advanced machine learning techniques have been applied to grain yield forecasting as seen in [67, 88, 81, 44, 123, 89, 82]. As mentioned, these articles focused on grain yield production, yet in terms of fresh produce there are very

limited articles found in literature. This number reduces dramatically for articles utilizing advanced machine learning for FP yield prediction [31].

Price forecasting has been an active research area in other domain such as electricity [34, 116, 106, 12], stock prices [54, 154, 62, 146, 8], real estate prices [49, 80, 78], etc. Prior to ML, statistical methods were employed in these domain but in recent times, advanced deep learning methods have been applied in some domains such as in the electricity price prediction [129, 20, 155, 63, 72], stock price prediction [139, 107, 76, 151] and real estate price prediction [48, 142, 64, 133]. Despite having a diverse and vast application where research is being performed, not much work is conducted in FP price prediction.

In this work, forecasting is used as a tool to measure the imputation performance due to the absence of the actual values required for the application of all popular performance measures listed in Section 2.6. Therefore, the impact of imputation on forecasting is gauged and used as the imputation performance indicator; the lower the forecasting error after imputation vs. before it, the higher the imputation performance. In pursuit of this several models for fresh produce yield and price forecasting are built. Those models cover the gaps in literature by forecasting FP prices in addition to yield, using compound DL models capable of learning complex relations.

2.8 Conclusion

The chapter covered the FP procurement and the unique challenges that entails it. It then went on to review the work which is being done in FP yield and price modelling.

The techniques used for imputation and forecasting are being described in details. Since the research deals with time series, different type of time series are being explored. This is followed by the traditional machine learning models are reviewed. After which deep learning models and the adjustments that are required to work with our time series are explained. Ensemble learning for improving model performance is reviewed next and then common evaluation metrics that are usually being employed in time series are illustrated. Finally, the research work related to this thesis is being outlined and the voids being filled by this research is being highlighted.

Chapter 3

Proposed Solution

3.1 Introduction

Tackling the problem of missing values requires careful consideration of different aspects. These aspects are covered in the following sections:

- Data: since the analysis of all the models is based on the data, it is imperative that it should be extracted correctly. Especially utilized weather data which require additional consideration since it is pre-processed in accordance to the length of the prior weather period for yield/price.
- Imputation Models: Different architectures for various imputation models are developed. The models range from statistical to deep learning and hence they cater the spectrum of complexity. The deep learning models are split into simple and compound DL models which are further improved by ensemble techniques.
- Forecasting Models: Various novel forecasting architectures are utilized. These forecasting models are used to gauge the impact of imputation on forecasting performance.
- Evaluation Metrics: It is very crucial to come up with the appropriate metric to evaluate the models in order to ensure that the models selected are operating correctly and optimally. Therefore, in order to gauge the performance of the imputation models, the mean squared error is used, whereas for the forecasting models, an aggregated measure (AGM), which combines mean squared error, mean absolute error and R^2, is used.

This chapter provides a detailed description of the aforementioned work aspects.

3.2 Datasets

Different datasets are used in building and testing the models proposed in this work. Detailed descriptions of these datasets are provided in the following sections.

3.2.1 California Weather Data Set

Since the majority of the strawberries are imported from California, consideration of this region's weather is critical and important. The weather data is collected from the main station in California, Santa Maria. The reason behind the selection of this specific station is the fact that more than 80% of the strawberry production comes from that station. The data is obtained from California Irrigation Management Information System (CIMIS) [120]. CIMIS records hourly, daily and yearly weather data. The data ranges from 2006 to 2019 and the obtained data has some missing values which are imputed using different methods. The experiments are carried out on the daily version of the data.

3.2.2 California Yield and Price

The strawberry yield and price are extracted from the California Strawberry Commission website [122]. The website has daily values which are used directly. The missing daily values are imputed via different advanced imputation models.

3.2.3 Corn Yield Dataset

The dataset involves a combined data of corn yield which was collected from the U. S. Department of Agriculture (USDA) that issues a monthly World Agricultural Supply and Demand Estimate (WASDE) report which includes projections for the Supply and Demand for various U. S. crops. The dataset included years from 2007 to 2018 with a total of 248 rows [1].

3.2.4 Oil Price Dataset

The crude oil price dataset consists of daily Brent and WTI oil prices [2]; these oil prices are univariate time series. The set is used to compare the proposed imputation models to one another and against the market models. The dataset ranges from 22/01/2008 to 22/08/2018. The dataset has 30% missing values which are filled using different imputation models and then the effectiveness of these models is gauged using forecasting models. For Brent, the forecasting is assessed in four different horizons: one day ahead, two days ahead, three days ahead and one week ahead whereas for WTI the forecasting model is assessed from one to twenty days ahead.

3.2.5 Complete Synthetic Time Series

Each time series exhibits different characteristics; hence it is difficult to generalize one imputation method to fit all types of time series. Therefore, four commonly used time series: trend, seasonal, combined and random time series are generated and examined to find the best imputation method for each.

3.2.5.1 Trend Time Series (T)

A trend time series moves in a simple linear fashion. The trend shows the general tendency of the data to increase or decrease during a long period of time. The increase or decrease does not need to be in the same direction throughout the given period of time [153]. Figure 3.1 shows the created trend time series.

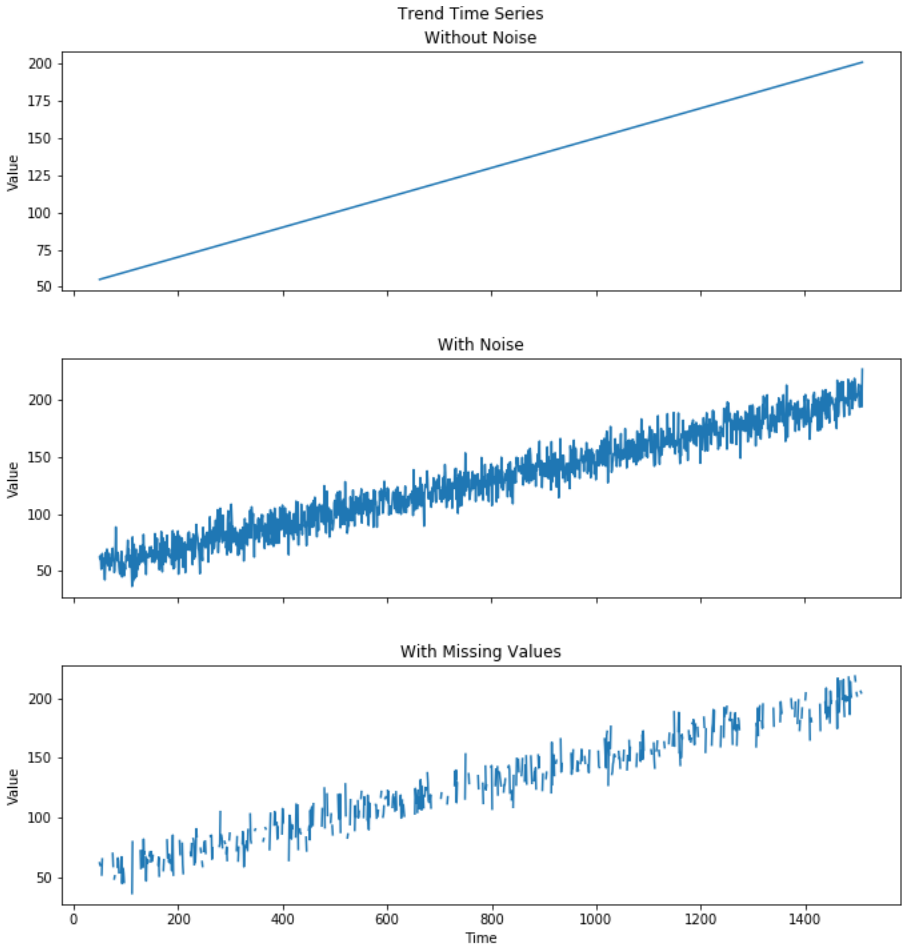

Figure 3.1. Trend time series.

3.2.5.2 Seasonal Time Series (S)

A seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week). Seasonality is always of a fixed and known period. Hence, seasonal time series are sometimes called periodic time series [50]. Figure 3.2 shows the seasonal time series.

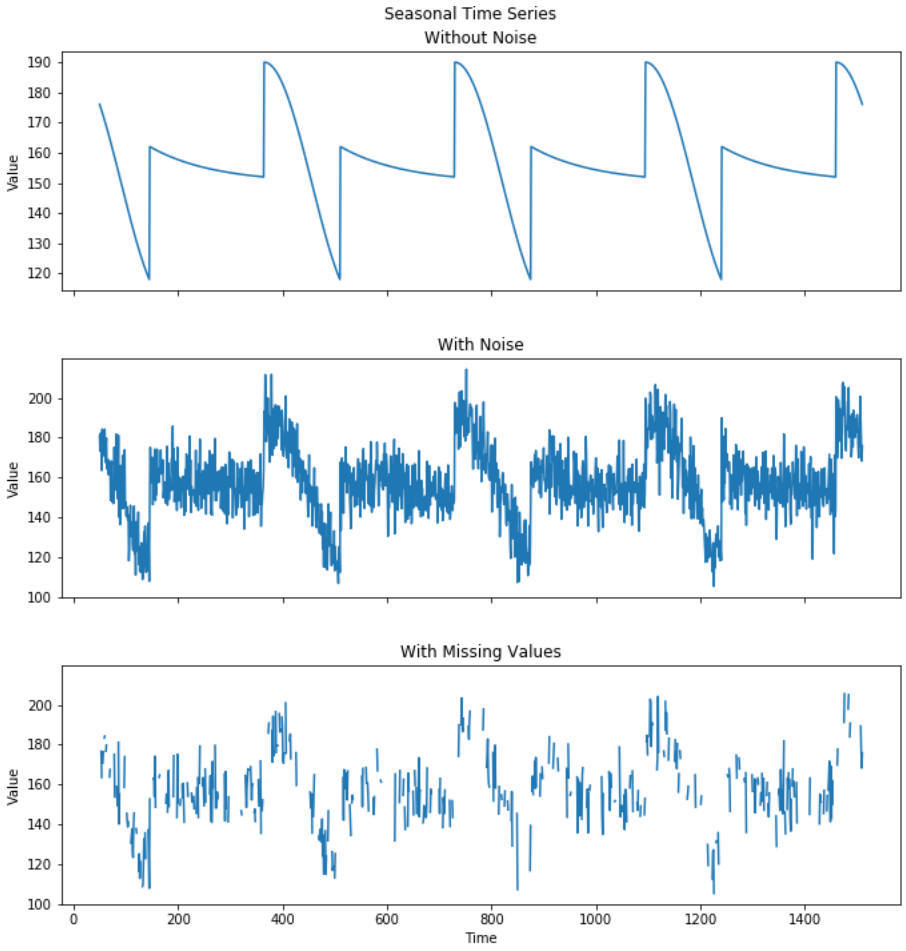

Figure 3.2. Seasonal time series.

3.2.5.3 Combined Time Series (T&S)

Many time series data contain the trend and seasonal patterns as its basic components. Trend and seasonality are very commonly encountered in economic and business time series [43]. Figure 3.3 shows the combined time series utilized in this work.

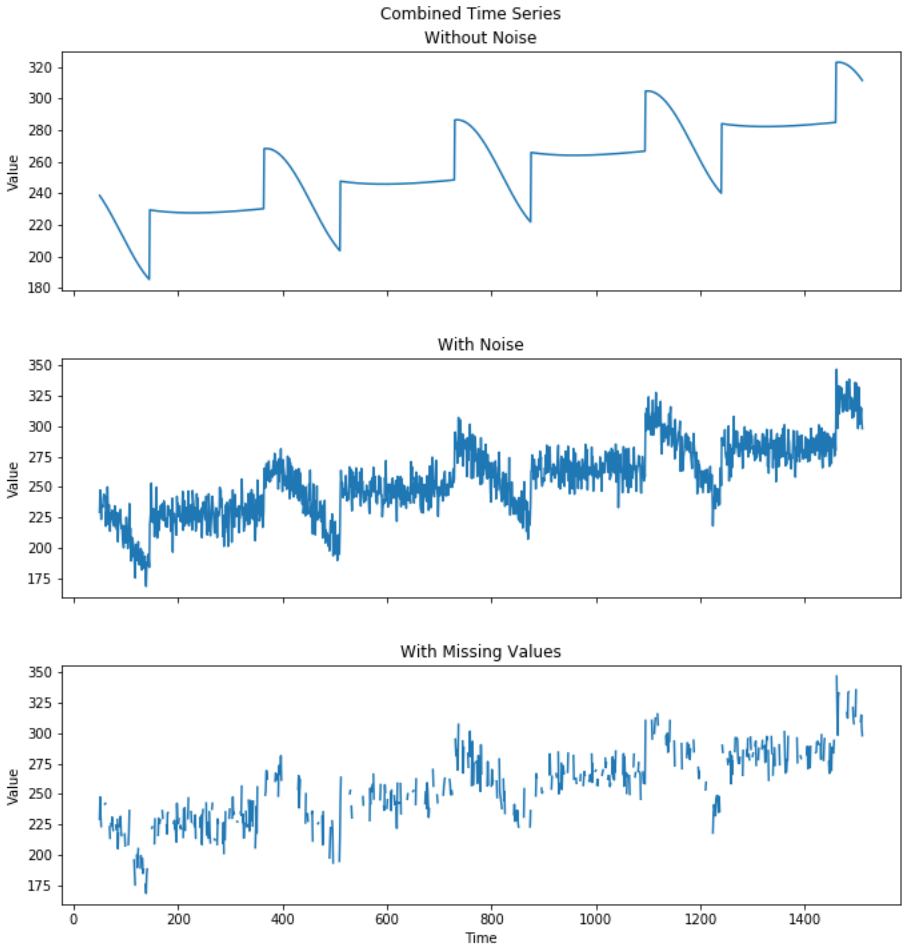

Figure 3.3. Combined (T&S) time series.

3.2.5.4 Random Time Series

A random time series is a time series that does not fall in any of the aforementioned categories; as illustrated in Fig. 3.4.

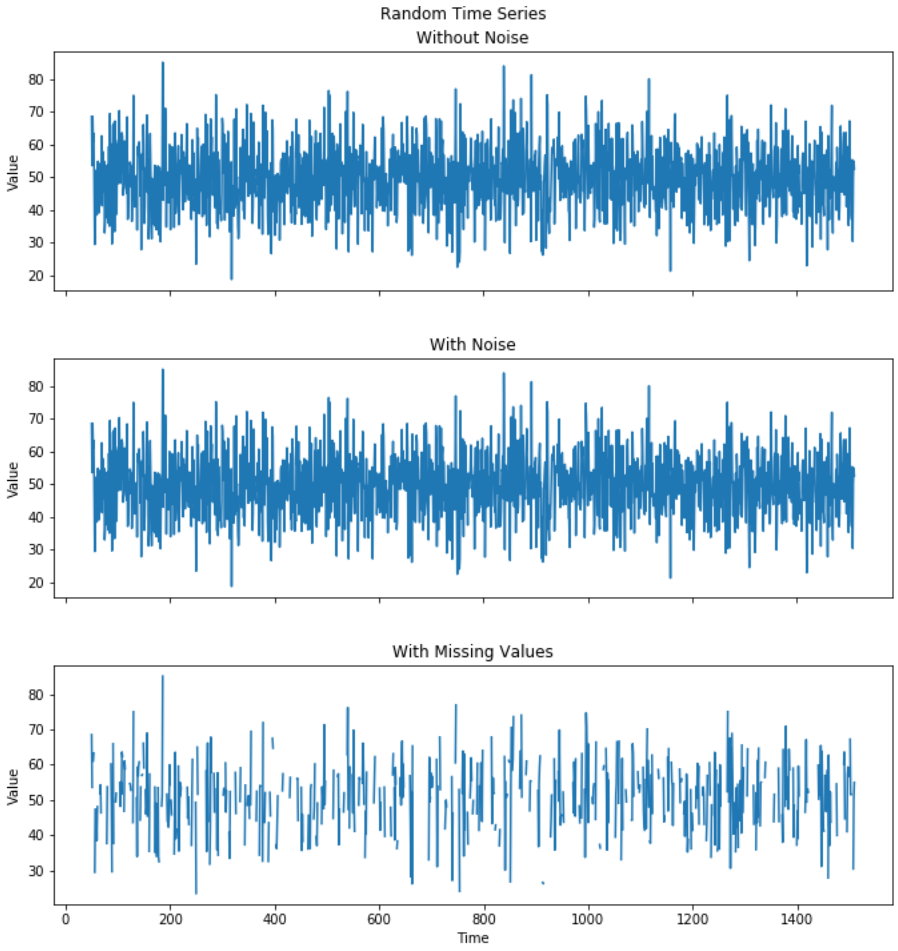

Figure 3.4. Random time series.

3.3 Imputation Framework

The proposed model is inspired by the work of Ma et al. [79]. The model is an RNN or its variant (LSTM or GRU) which has a learned linear combination of the previous states. The neural network is trained on the data and a vector, called Residual Sum Vector (RSV), is added to it. RSV is the weighted sum of the model's previous hidden states and is called RSV in analogy to the residual connections in a ResNet. The learned memory weights are then used to impute the missing values in the dataset.

The network operates in such a way that it learns and predicts simultaneously. Whenever the network finds the input variable present, it learns to regress it, and when the values are missing, they are filled based on this regression. This framework imputes randomly missing values in the dataset.

The considered time series are of T length as $X = \{x_1, x_2, ...x_T\}$ where $x_i \in \mathrm{R}^n$. An indicator vector of missing values is introduced, since the considered time series are incomplete, $M = \{m_1, m_2, ...m_T\}$ where $m_t \in \{0,1\}^n$. The m_{it} indicates whether the value is present at the ith position or not. $m_{it} = 0$ indicates that the value is present, while $m_{it} = 1$ portrays otherwise.

The RSV is deployed in the neural network in order to incorporate the past information from the hidden states. The output of RSV at time t is an RSV denoted by $r_t \in \mathrm{R}^m$ and its dimension is same as h_t. r_t is defined as:

$$r_t = \begin{cases} f(h_t), & \text{if } t = 1. \\ f(h_t + g(W_r, r_{t-1})), & \text{if } t = 2, 3, ..., T \end{cases} \tag{3.1}$$

here, g and f are the vector valued functions, $W_r \in \mathrm{R}^{m \mathrm{x} m}$, and $h_t \in \mathrm{R}^m$ denotes the output of the neural network hidden layer at time t. The next input is approximated using:

$$z_{t+1} = W_{imp} r_t \tag{3.2}$$

z_{t+1} is trained to approximate x_{t+1} if it is present and is used to impute x_{t+1} if it is missing.

The training of the model is done using back propagation of error. The process runs in two specific cases, approximation and imputation. If the next input x_{t+1} is present in the dataset then the output z_{t+1} is trained to approximate the input x_{t+1}. In the case where x_{t+1} is a missing value then z_{t+1} is directly copied to x_{t+1}. The approximating loss, L_{t_approx}, at each time step t, is the squared error loss between the original and approximated value, according to the existence of input x_t.

$$L_{t\_approx}(z_t, x_t) = \| \ (z_t - x_t \circ \neg m_t) \ \|_2^2 \tag{3.3}$$

here, the $\neg m_t$ is masking off the missing data from the approximation loss.

The overall training loss has two components, the total approximating loss term (3.4) and the task related loss term (3.5), which are as follows:

$$L_{total\_approx} = \sum_{k=1}^{N} \sum_{t=1}^{T-1} L_{t\_approx}(z_{t+1}^{(k)}, x_{t+1}^{(k)}) \tag{3.4}$$

$$L_{total\_target} = \sum_{k=1}^{N} L_{target}(d^{(k)}, y_t^{(k)}) \tag{3.5}$$

The superscript k here denotes the kth sample of the time series. The $d^{(k)}$ and $y_t^{(k)}$ denote the task related target and the output of the kth value. The total training loss, L_{total}, is obtained by combining (3.4) and (3.5).

$$L_{total} = L_{total\_approx} + \lambda_{target} L_{total\_target} \tag{3.6}$$

where λ_{target} is a coefficient weighing the importance of the task loss. This loss function is optimized by the Back Propagation Through Time algorithm. The overall working of the framework is illustrated in Fig. 3.5.

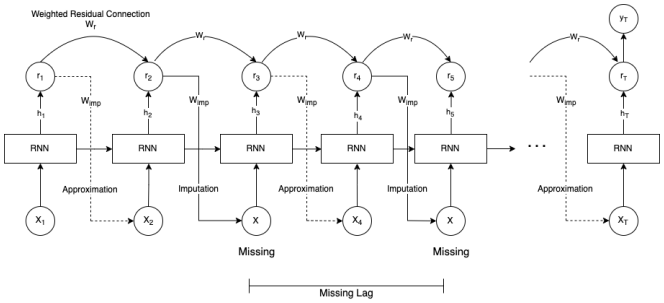

Figure 3.5. Illustration of the imputation framework [79].

3.4 Non-Machine learning (Non-ML) Models

Various types of models are built to tackle the issue of imputation. They are highlighted in the following subsections.

3.4.1 Conventional Models

1. *Complete Deletion Case (CCA):* This is the case where the rows having missing values are deleted and hence not considered while feeding the data into the model.

2. *Last observed value carried forward (LOCF):* In this model, the previously observed values are used in place of the missing values.

3.4.2 Statistical Models

1. *Mean Imputation:* The missing values are replaced by the mean of the feature.
2. *Median Imputation:* The missing values are replaced by the median of the feature.
3. *Mode Imputation:* The missing values are replaced by the mode of the feature.

3.4.3 Nondeep Machine Learning Models

For the nondeep learning models, the following models are implemented:

1. Linear Regression
2. Support Vector Regression
3. K-nearest Neighbours
4. Gradient Boosting Regression

For all the aforementioned models, default parameters are used except for Gradient Boosting Regression where the "huber" loss function [87] is utilized for optimal results.

3.4.4 Deep Learning Models

Deep learning models have the ability to learn complex patterns in the data and they also continuously improve with the increase in data size [52]. Due to these abilities, deep learning models are considered as part of the proposed solution. The deep learning models are categorized into simple, compound and attention-based compound deep learning models.

3.4.4.1 Simple Deep Learning Models

The simple deep learning models proposed in this work are: RNN, LSTM and GRU.

1. *RNN*: The architecture of the RNN model starts with one layer of RNN with 100 units. The MSE loss function is used and the optimizer is Adam [68].

2. *LSTM*: The architecture of the LSTM model starts with one layer of LSTM with 100 units. The MSE loss function is used and the optimizer is Adam [68].

3. *GRU*: The architecture of the GRU model starts with one layer of GRU with 100 units. The MSE loss function is used and the optimizer is Adam [68].

3.4.4.2 Compound Deep Learning Models

The compound deep learning models utilized in this thesis are explained as follows:

1. *Residual GRU*: Single layer of GRU with residual unit attached to it is used in this model. Residual network connects input in front of the layer directly to the output layer as illustrated Figure 3.6.

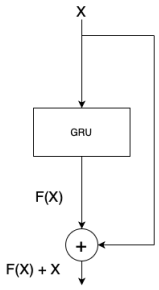

Figure 3.6. Residual Network.

2. *LSTM-GRU*: This is a special type of neural network. It consists of two hidden layers where one layer is LSTM followed by GRU layer. This type of model produces promising results in stock price prediction as demonstrated in [99]. A total of 100 nodes are used for each layer. MSE is used as the loss function and the utilized optimizer is Adam [68].

3. *Deep GRU*: The model is a deeper version of simple GRU. It consists of 2 layers of GRU. A total of 50 nodes are used for each layer. MSE is used as the loss function and the utilized optimizer is Adam [68].

4. *LSTM- Deep GRU*: The model is composed of an additional GRU layer with the LSTM-GRU configuration. A total of 100 nodes are used for each of the layer. MSE is used as the loss function and the utilized optimizer is Adam [68].

5. *Deep LSTM-GRU*: The model is a deeper version of the LSTM-GRU model. It consists of 4 layers under the configuration LSTM-GRU-LSTM-GRU. A total of 50 nodes are used for each layer. MSE is used as the loss function and the utilized optimizer is Adam [68].

6. *Encoder-Decoder*: The encoder-decoder architecture is a neural network design pattern. As shown in Figure 3.7, , the architecture is partitioned into two parts, the encoder and the decoder. The encoder's role is to encode the inputs into a state which often contains several tensors. Then that state passes into the decoder to generate the outputs. The model has shown great performance in the field of natural language processing [147].In time series, the input sequence is the historic data whereas the output sequence is the forecasting. In this research, 3 layers of GRU are utilized for both encoder and decoder each with 35 nodes. MSE is used as the loss function and the utilized optimizer is Adam [68].

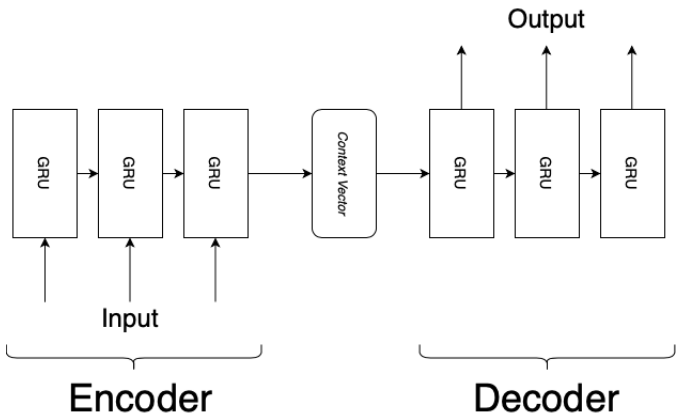

Figure 3.7. Encoder-Decoder Network

3.4.4.3 Attention Based Models - SeriesNet

The architecture of the network is illustrated in Section 2.4.7. This section will highlight all the features and modification that are being implemented in the architecture to make more robust.

The architecture of SeriesNet consists of 2 or more networks. One network is of dilated causal convolution and other ones are variety of the networks. The distinguishing factor of this network is that it is has attention module in it thereby, improving its forecasting accuracy.

The dilated causal convolution is composed of 7 layers of dilated convolution operations followed by a 1D convolution layer. The filters used in the dilated convolution operation are 32 while the dilation rate is increased 2 times for each layer. The output of the 1D convolution layer is passed through the attention layer after which it is flattened and fed to single neuron dense layer.

Outputs from all of the networks are concatenated and passed through Relu activation function to give the overall output. The optimizer utilized is Adam [68] and the loss function of MSE is used.

3.4.4.4 SeriesNet with LSTM

In this model, along with dilated causal convolution, LSTM is employed as another network. The LSTM network is a two layer network between which attention module is introduced. The output at the end is flattened and fed to single neuron dense layer. The number of layers and neurons alter as per the application. The architecture of the model can be summarized in the block diagram below.

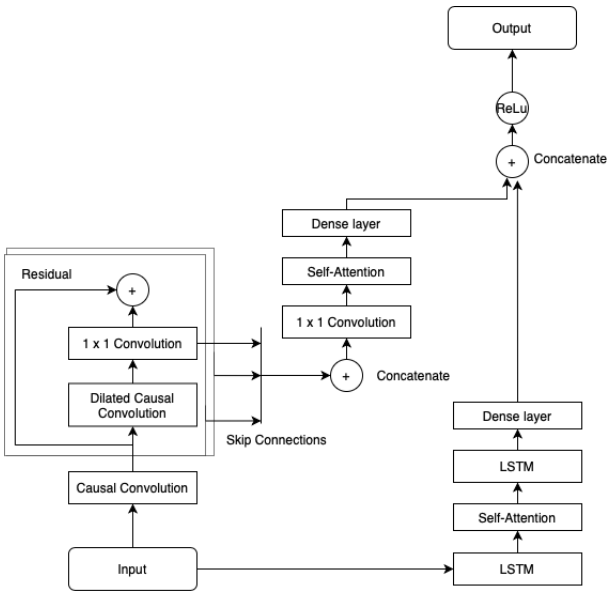

Figure 3.8. SeriesNet with LSTM

3.4.4.5 SeriesNet with GRU

In this model, GRU is used alongside dilated causal convolution. The GRU network is a two layer network between which attention module is introduced. The output at the end is flattened and fed to single neuron dense layer. The number of layers and neurons alter as per the application. The architecture of the model can be summarized in the block diagram below.

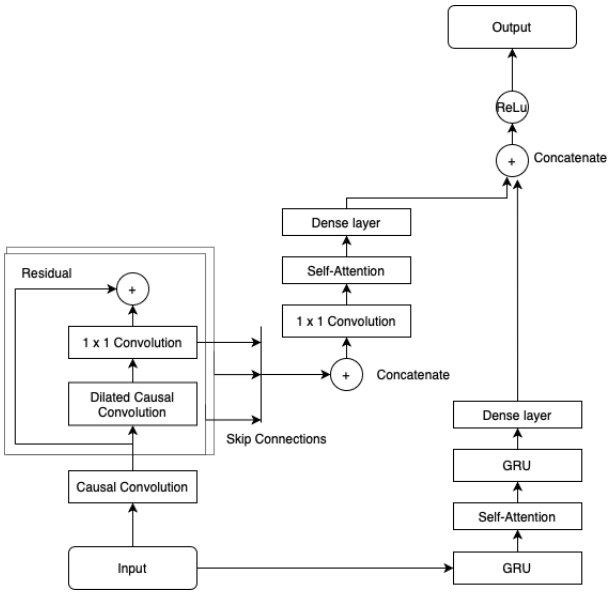

Figure 3.9. SeriesNet with GRU

3.4.4.6 SeriesNet with CNN-LSTM

The model consists of a network of CNN-LSTM along with the network of dilated causal convolution. The CNN-LSTM network is a 4 layer convolutional followed by 2 layers of LSTM network after which attention module is introduced. The output at the end is flattened and fed to single neuron dense layer. The convolutional layer has 12 filters, stride of 1 and a kernel size of 3. Whereas, LSTM network has 100 nodes in each layer. The architecture of the model can be summarized in the block diagram below.

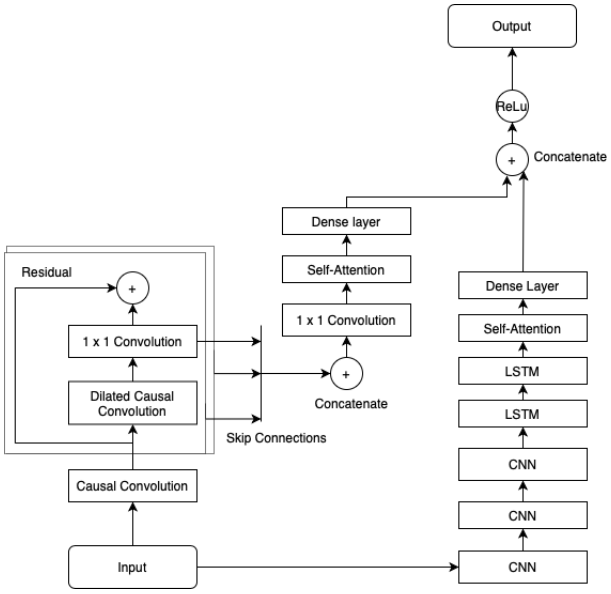

Figure 3.10. SeriesNet with CNN-LSTM

3.4.4.7 SeriesNet with ConvLSTM

The model uses a network of ConvLSTM along with dilated causal convolution. The ConvLSTM network is a two sets of 2D ConvLSTM layers with 128 filters and kernel size (1, 3) each immediately followed by batch normalization. A self- attention layer with sigmoid activation follows after that. The output at the end is flattened and fed to single neuron dense layer. The architecture of the model can be summarized in the block diagram below.

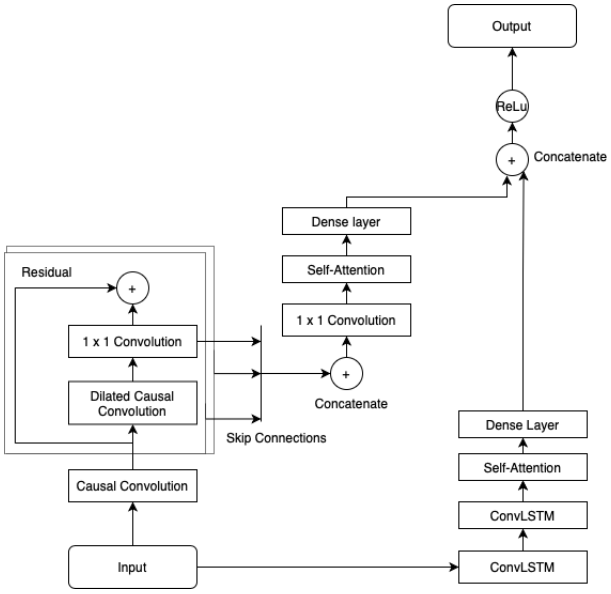

Figure 3.11. SeriesNet with ConvLSTM

3.4.4.8 SeriesNet with CNN-LSTM and ConvLSTM

The architecture consists of 3 networks. One network is of dilated causal convolution, second one is the network of ConvLSTM and the third network is of CNN-LSTM. The ConvLSTM network is a two sets of 2D ConvLSTM layers with 128 filters and kernel size (1, 3) each immediately followed by batch normalization. A self- attention layer with sigmoid activation follows after that. The output at the end is flattened and fed to single neuron dense layer. The CNN-LSTM network is a 4 layer convolutional followed by 2 layers of LSTM network after which attention module is introduced. The output at the end is flattened and fed to single neuron dense layer. The convolutional layers have 120 filters, stride of 1 and a kernel size of 3. Whereas, LSTM network has 100 nodes in each layer. The architecture of the model can be summarized in the block diagram below.

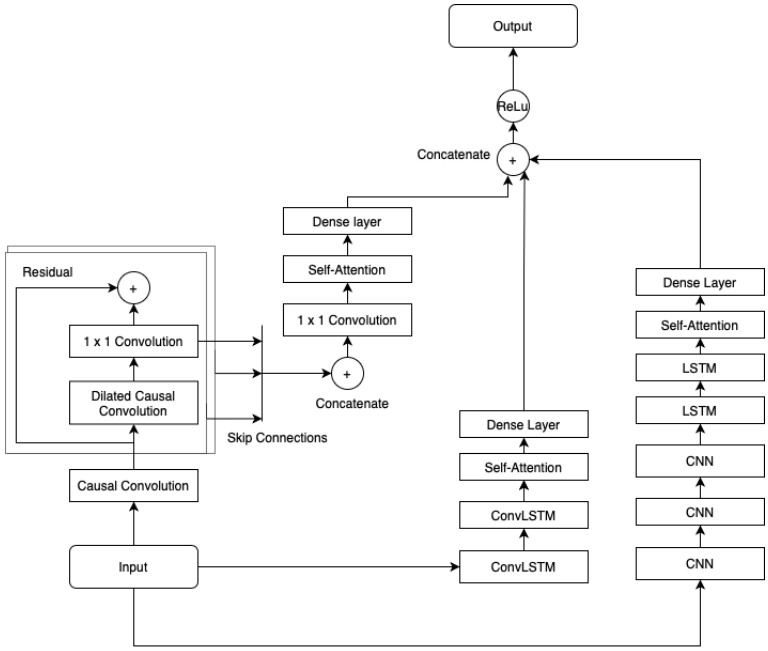

Figure 3.12. SeriesNet with ConvLSTM & CNN-LSTM

3.4.4.9 SeriesNet with CNN-LSTM and GRU

The architecture consists of 3 networks. One network is of dilated causal convolution, second one is the network of CNN-LSTM and the third network is of GRU. The network configuration is the same as mention in the previous architectures. The architecture of the model can be summarized in the block diagram below.

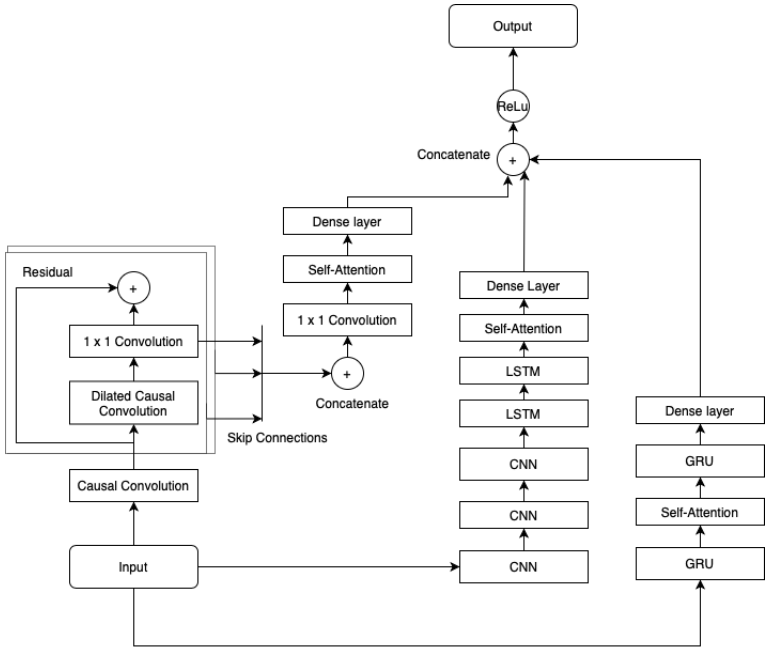

Figure 3.13. SeriesNet with CNN-LSTM & GRU

3.4.4.10 SeriesNet with ConvLSTM and GRU

The architecture consists of 3 networks. One network is of dilated causal convolution, second one is the network of ConvLSTM and the third network is of GRU. The network configuration is the same as mention in the previous architectures. The architecture of the model can be summarized in the block diagram below.

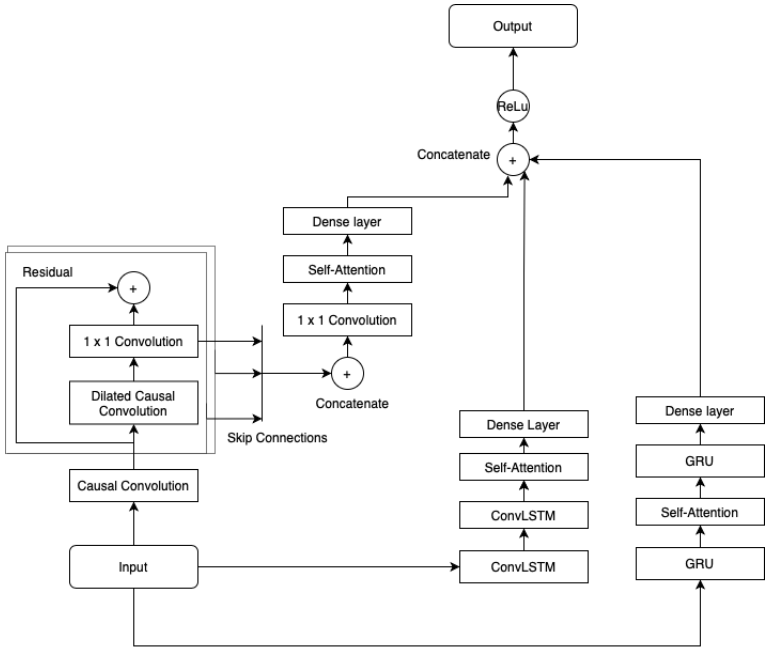

Figure 3.14. SeriesNet with ConvLSTM & GRU

3.4.5 Ensemble Technique

For imputation as well as for forecasting, an ensemble is built for the top two performing models. They are selected primarily on the basis of the experiments results. It is found that usually creating an ensemble of the top two models helps in imputation whereas it doesn't give a significant improvement in forecasting. The techniques used are those mentioned in chapter 2, voting regression and stacking. The stacking algorithm is comprised of Linear regression, support vector regression (SVR) and Gradient Boosting regression (GBR).

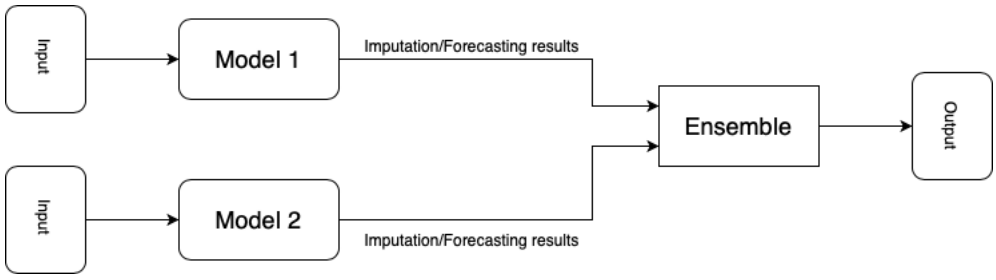

Figure 3.15. Ensemble architecture

3.5 Evaluation Metric

All the evaluation metrics are described in Section 2.6 along with their pros and cons. For the imputation phase, the mean squared error is used to gauge the effectiveness of the model. However, gauging the effectiveness of forecasting is more complicated. Since each metric has its own merit and demerit, more than one metric is used. In order to decide one best model, one aggregated performance value is needed for each model. Hence, a new error metric is proposed for forecasting which is the Aggregate Error Measure (AGM). This aggregated measure is developed to combine the three utilized popular metrics: RMSE, MAE and R^2. It is calculated by averaging RMSE (square root of MSE) and MAE and multiplying the result by (1-R^2). The factor (1-R^2) represents the portion of variance that is not captured by the model. Since the R^2 is a dimensionless constant, the final error maintains the same scale as the dependent variable. The selection of the forecasting error and benchmarking is done via AGM.

3.6 Model Tuning

One of the most difficult tasks in setting up the deep learning model is tuning the hyper-parameters. Amongst all the hyper-parameters, learning rate is the most important one [140]. It is the step size that the algorithm follows as it moves towards the global/local minimum. This would imply that in case the learning rate is small, the convergence of the algorithm will be slow [140]. On the contrary, in case a large learning rate is set, it will be stuck in a local minimum [140]. Therefore, model tuning is important in terms of making the model generalizable; i.e. it neither overfits nor underfits. The term overfit refers to a situation where the model learns the training data and is not able to find the pattern within that data. This would mean that the model has a poor accuracy with test data. Overfitting can happen due to having a small dataset, complex model, etc. In order to solve the model complexity issue, regularization is usually done which makes it difficult for the model to learn and hence enabling it to find the pattern in the data. Underfitting is the situation where the model is neither able to generalize the training data nor the test data. Finding the balance between both scenarios is the aim of every supervised learning task.

In order to select the parameters of the models the following steps were done.

- Model checkpoint was used. Model checkpoint saves the weights of the models based on the set condition. In this work, the minimum loss of validation data is set as

the condition for model checkpoint. This is helpful because it tackles the overfitting problem that can cause due to high number of epochs.

- Manual iterations were used to determine learning rate, model size, optimizer algorithm, error function, regularization parameter, etc.

3.7 Conclusion

This chapter outlines the various datasets that are being utilized in the experiments carried out in this thesis. The datasets include the following:

- California weather data obtained from Santa Maria which comprises of 13 weather parameters.
- California strawberry yield and price dataset.
- Daily oil prices dataset.
- Corn yield dataset illustrating corn yield in USA.
- Synthetic time series dataset having complete time series data generated synthetically.

Next, details regarding the structure of the proposed models are outlined. The models were categorized into non-machine learning models, non-deep machine learning models and deep learning models. Finally, a new evaluation metric for choosing the prediction is proposed. The next chapter will highlight the details of the experiments and the results obtained by them.

Chapter 4

Experiments & Result Analysis

4.1 Introduction

The previous chapter covers the proposed solution for imputation and forecasting. It also encompasses the datasets, models and evaluation metrics used to derive the best solution. In this chapter, the experiments carried out are outlined. These experiments can be divided into two main categories which are imputation and forecasting. Experiments belonging to each are explained in details in the following sections.

4.2 Imputation Phase

The imputation phase follows the strategy in which first the methodology for imputation is selected, which is followed by the model exploration or selection and at the end, the models are improved by ensembling techniques. Each component of the strategy will be explained in details.

4.2.1 Methodology Selection

Since imputation is an active research topic, multiple models belonging to different domains are developed. Therefore, it is imperative for us to determine the optimal domain then move further. Hence the first step in the imputation roadmap deals with the determination of the best domain for imputation. In this experiment, three main domains of imputation

namely non-machine learning, non-deep machine learning and deep learning methods are explored. Although there exists a long list of models under each methodology, the focus is on the three most common models under each domain. Under the non-machine learning methodology, models like mean, mode, and median are used. The Linear Regression, Support Vector Regression and Gradient Boosting Regression models are used under the non-deep machine learning methodology. Finally, Linear Memory Vector Recurrent Neural Network (LIME-RNN), Linear Memory Vector Long Short Term Memory (LIME-LSTM), and Linear Memory Vector Gated Recurrent Unit (LIME-GRU) are used under the deep learning methodology. The comparative performance analysis is performed on one dataset for the three mentioned methodologies.

The missing values are generated randomly in a complete-time series dataset (with no missing values) of corn yield [1] and various models belonging to the mentioned methodologies above are used to impute the missing values in the dataset. The imputed values are then compared with the original values to deduce the method providing the most appropriate imputation results. Once the performance analysis of each model is obtained, the worst and best performing models are applied on the WTI oil price dataset [2], to validate the method and gauge its impact on prediction. For prediction a window of 20 days is used.

Hence the experiment is divided into two phases. The first phase deals with the comparative study between the imputation methodologies (non-machine learning, Nondeep Machine Learning and Deep Learning methodologies), while the second phase determines the impact of imputation on forecasting.

For imputation, the time series dataset on monthly corn yield, which had no missing values, is used. 50% random missing values are generated in the complete dataset, and aforementioned imputation techniques are used to impute those. The imputed values are compared with the original values, and the Mean Squared Error (MSE) is calculated between both. The block diagram in Figure 4.1 can be referenced for the overall imputation methodology.

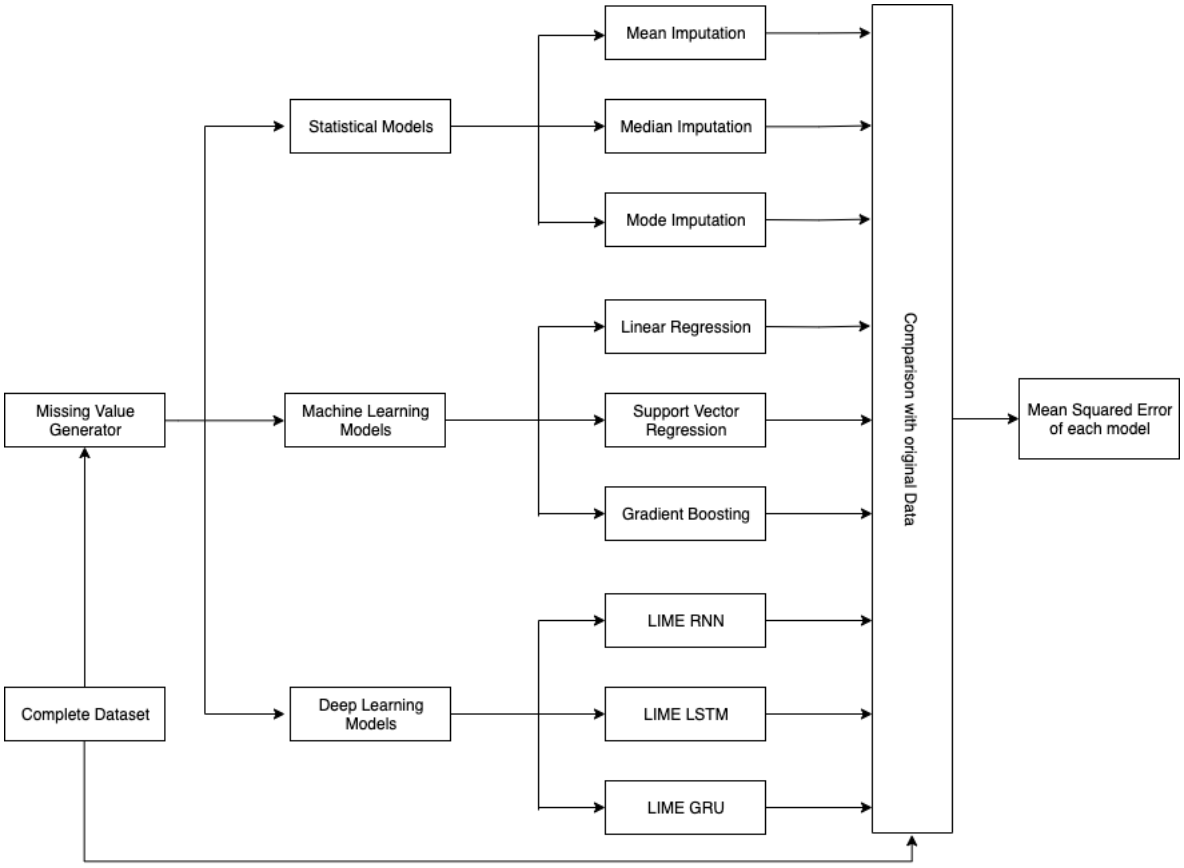

Figure 4.1. Imputation Methodology

After calculating the MSE for each imputation model, the worst and best-performing models are applied to impute the missing values in the oil price dataset. The resulting imputed datasets are used to forecast the oil prices. A 20 days ahead prediction is made using a single LSTM hidden layer with 100 nodes and 150 epochs. The metric used for prediction is AGM as explain in Section 2.6.

The results of imputation are summarized in Table 4.1 and Figure 4.5. A snapshot of these results can be viewed in Figure 4.2.

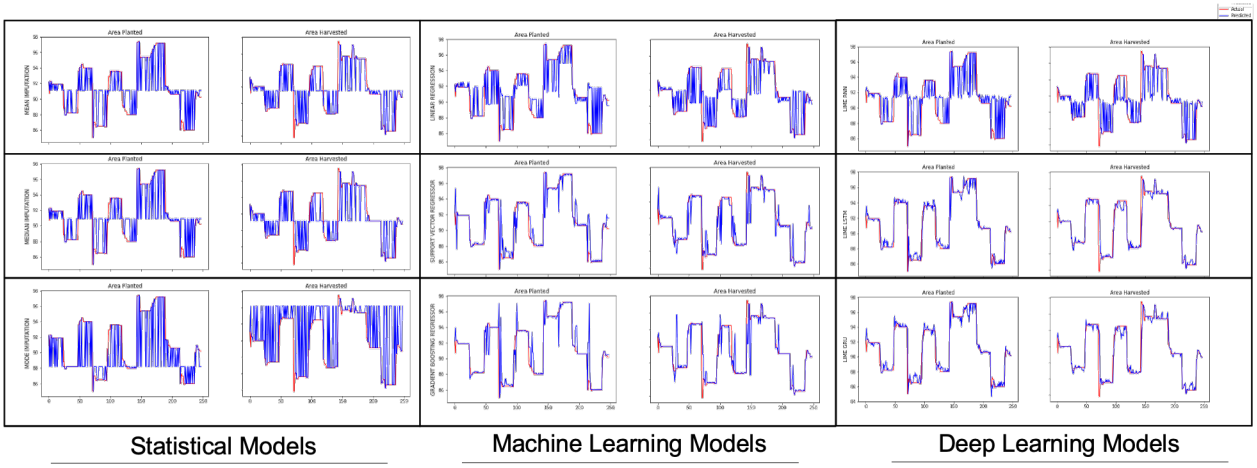

Figure 4.2. Snapshot of the results of the models.

The left hand side of Figure 4.2 illustrates the results from statistical models. As evident, the models do not provide satisfactory results. Amongst these three statistical models, mode imputation provides the worst performance results with the highest MSE value. The reason behind the poor performance is that the statistical models do not consider the dependencies in the time series. They compute the mean, median, or mode of the dataset and replace it with the missing value, thereby creating a bias in the dataset. As per the results, it is also observed that the mean and median imputations provide comparable results, which highlights the fact that both features (Area Planted and Area Harvested) follow almost the same distribution.

The middle section of Figure 4.2 shows the results of the machine learning models. It is observed that the machine learning models predict the missing values with greater accuracy than the statistical models. This is by virtue of their learning mechanism. Amongst the three models, Linear Regression has the worst results, and that is due to its inability to cater to any non-linearity in the dataset. Support vector regression and Gradient boosting regression provide similar results as they are able to consider the non-linearity in the dataset, thereby imputing the missing values with better precision.

The right side section of Figure 4.2 shows the imputation results of the LIME framework. All the models, except for LIME-RNN, achieve a higher accuracy amongst all the examined imputation models, as evident from Table 4.1. The degraded performance of LIME-RNN is due to the lack of a memory unit which is present in LIME-LSTM and LIME-GRU. This hinders LIME-RNN in catering to the dependencies between the miss-

ing value and the previous values. Additionally, it is also observed that LIME-GRU has a lower MSE than LIME-LSTM. This is due to the more straightforward structure of GRU, which allows the error to backpropagate without vanishing too quickly as it bypasses multiple temporal steps [25].

Table 4.1
Results of imputation by the models employed.

Domains	Models	MSE
Statistical Methods	Mean Imputation	*3.85*
	Median Imputation	*3.99*
	Mode Imputation	*15.13*
Machine Learning Method	Linear Regression	*3.56*
	Support Vector Regression	*1.11*
	Gradient Boosting Regression	*1.45*
Deep Learning Method	LIME-RNN	*3.5*
	LIME-LSTM	*0.74*
	LIME-GRU	*0.64*

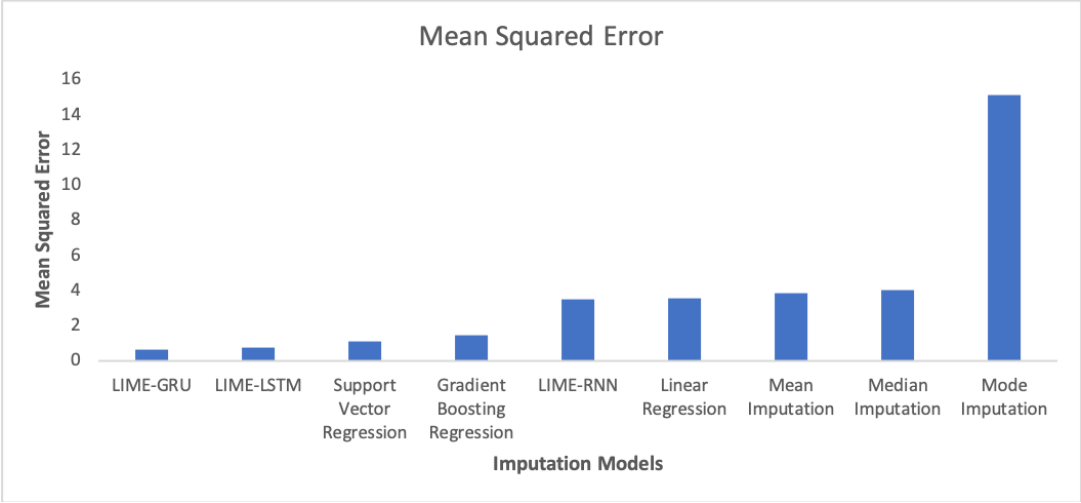

Figure 4.3. Comparison between the different imputation methods that are being used.

To further validate the results, the worst and best-performing imputation models are applied to the WTI oil price dataset [2], and their impact on its prediction capabilities

is observed. To compare the performance of learning models with different forecasting horizons, forecasting from one day to twenty days ahead is done. Moreover, 80% of the data points in the dataset are used for training, while the remaining 20% are used for testing. The results can be viewed in Table 4.2 and Figure 4.4. As can be observed, the imputation via LIME-GRU gives a better prediction results. Overall, there is a 39% improvement in the prediction errors when LIME-GRU is used as a imputation model as opposed to mode imputation. This validates the results found previously.

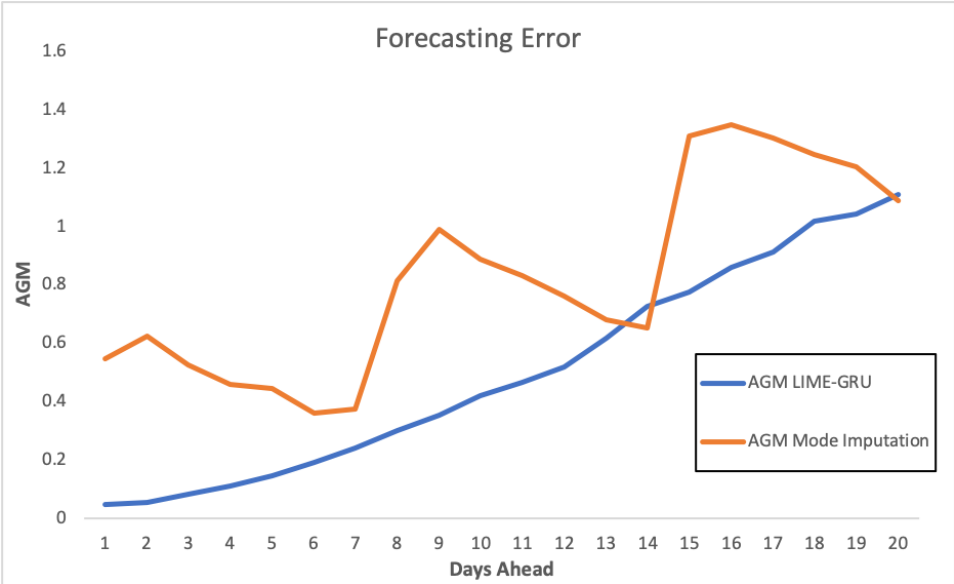

Figure 4.4. Illustration of AGM on each predictive step.

Table 4.2
AGM and overall AAGM for each imputation method.

Steps Ahead	AGM LIME-GRU	AGM Mode Imputation
1	*0.05*	*0.55*
2	*0.05*	*0.62*
3	*0.08*	*0.53*
4	*0.11*	*0.46*
5	*0.14*	*0.44*
6	*0.19*	*0.36*
7	*0.24*	*0.37*
8	*0.30*	*0.81*
9	*0.35*	*0.99*
10	*0.42*	*0.89*
11	*0.46*	*0.83*
12	*0.52*	*0.76*
13	*0.62*	*0.68*
14	*0.73*	*0.65*
15	*0.78*	*1.31*
16	*0.86*	*1.35*
17	*0.91*	*1.30*
18	*1.02*	*1.25*
19	*1.04*	*1.21*
20	*1.11*	*1.09*
AAGM	***0.50***	***0.82***

All in all, a comparative study between the three methodologies: non-machine learning, non-deep machine learning, and deep learning shows that the deep learning methodology provides the most accurate imputation results compared to the other examined approaches. This result is further validated via a predictive model. However, the only drawback in the utilization of the deep learning models is the computational resources required. Since deep learning models by virtue of it architecture requires a lot of resources whereas, machine learning or statistical model don't require that much of the resource.

4.2.2 Model Selection

Once the methodology is decided in Section 4.2.1, further exploration is done to find the most accurate imputation models within that methodology . Therefore, in this experiment, four types of time series are considered: (1) Trend (T) time series, (2) Seasonal (S) time series, (3) Combined Trend and Seasonal (T&S) time series, and (4) Random (R) time series and seven different DL models are applied to impute the missing values in each of these time series. The utilized imputation models range from the most basic to the highly complex ones, thereby allowing coverage of the spectrum of complexity. The performance

analysis of the imputation methods helps in finding the best model for each time series type. The reason for determining optimal imputation for each time series type is because every time series carry its unique characteristic and therefore, generalizing one model for all is not possible.

The utilized imputation models are: (1) Simple DL models including the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU). (2) Compound DL models which involve the Residual GRU, LSTM-GRU, Deep GRU, LSTM-Deep GRU, and Deep LSTM-GRU. The models are also compared against a non-DL model that uses linear function for interpolation. In order to gauge the effectiveness of the model with one value, AGE is used to determine the top two imputation models for each tested time series type; AGE is calculated by comparing imputed values with actual ones. The results are summarized in Table 4.3 and are depicted in Figure 4.5.

Table 4.3
AGE of each imputation method against each time series. The bold figures illustrate the top 2 performing models for each type.

Time Series Type	Measure	Imputation Model							
		Function	*Simple DL*		*Compund DL*				
		Linear	LSTM	GRU	Deep GRU	LSTM-GRU	Deep LSTM-GRU	Residual GRU	LSTM - Deep GRU
Trend	AGE (RMSE + MAE)/2	*11.71*	*10.60*	*9.92*	*9.94*	***9.89***	*10.05*	***9.86***	*10.44*
Seasonal		*11.55*	*10.60*	*10.82*	*11.20*	*10.75*	*11.45*	***10.17***	***10.54***
Combined		*11.50*	*10.94*	***10.73***	*11.64*	*10.83*	*12.45*	***10.75***	*10.98*
Random		*11.15*	*8.91*	*8.91*	*8.91*	***8.90***	*8.91*	*8.91*	***8.91***

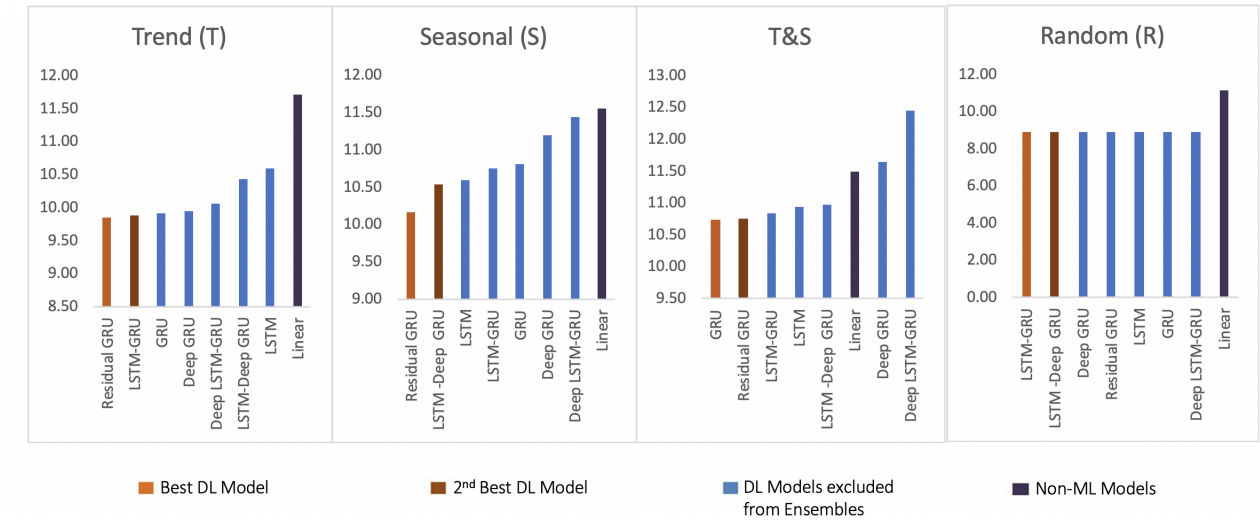

Figure 4.5. Performance comparison of the utilized imputation methods.

It can be seen that in both Trend and Seasonal time series the Residual GRU Network provides the best imputation results. This is because the network introduces the non-linearity in the system which enables it to predict and impute values close to the actual ones. Compared to the Residual GRU Network, more complex models tend to deteriorate the results as they are unable to predict the values with high precision. Similarly, simple DL models, i.e. LSTM and GRU, are unable to cater the non-linearity of the time series since they are rudimentary models.

In the combined (T&S) time series, it is evident that GRU based neural networks are in general the most suitable choice for imputation. Since the combined time series incorporate the characteristics of both Trend and Seasonal time series, GRU based neural networks are found to be optimal in both.

It is also observed that in this case the simple GRU network is the best performing model compared to the other models. The optimal results obtained by the GRU are due to the simpler structure that enables it to take into account the non-linearity. Furthermore, the GRU performs better than LSTM because GRU allows the errors to back-propagate without vanishing too quickly as it bypasses multiple temporal steps [25].

In the Random time series, almost all the models are found to be suitable for imputation. Each of the models has almost the same AGE value and thus, any of them can be used. Such results are found due to the randomness in data. Since it is a random time series, it

consists of all the components of the time series, hence all the models produce comparable results.

4.2.3 Models Ensemble

Once the most appropriate imputation model for each time series is determined from Section 4.2.2, the next step of the process is to further improve the performance by employing ensemble techniques. Since it is the continuation of the previous experimentation, the same synthetic time series dataset used in Section 4.2.2 is used. Two different types of ensemble techniques are utilized which are Voting Regressor and Stacking ensemble.

In voting regressor, the top two performing models are averaged out whereas in stacking ensemble, 3 machine learning (ML) ensemble techniques are utilized. These ML techniques include Linear Regression (LR), Support Vector Regressor (SVR), and Gradient Boosting (GB) ensembles.

The application of voting regressor is straightforward as it is the simple average of the output of the top two performing imputation models. Conversely, Stacking ML ensemble is a bit is more complicated. To apply the ML ensemble, an additional 10% of the values from the original dataset, which originally had 50% missing values, have to be masked. The top 2 imputation methods are applied on the resulting dataset. From imputation results of these DL models, the 10% masked values which are imputed are stacked along with their original values. The imputed values are considered as the input, while the actual values are considered as the output for training the ML ensemble model. Masking of the values is done because ML models are supervised models that require both the output along with its corresponding input to train itself. After the model is trained, it is used to impute the 50% missing values from the original dataset. The ensemble techniques can also be illustrated in the Figure 4.6.

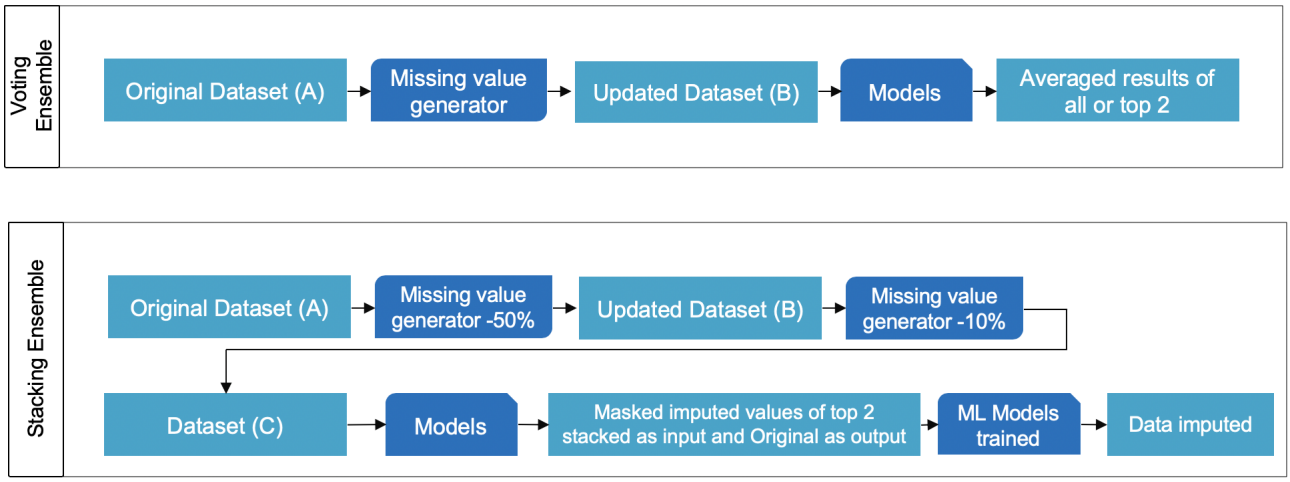

Figure 4.6. Ensemble Technique block diagram.

In terms of parameters, LR model uses the default parameters, SVR uses linear as its kernel and huber function is used for calculating the loss in GB ensemble [87].

The results based on the AGE metric are illustrated in Figure 4.7 and Table 4.4 for all four tested types of time series. As evident from Table 4.4 and Figure. 4.7, the ML ensemble did not provide satisfactory results. This is due to the masking of additional 10% of the input data to the ML ensemble which is needed to train the ML model. This in turn means that the DL models have a greater portion of missing values to impute, which results in higher AGE. However, voting ensemble did outperform all the ensemble as well as DL imputation models.

Table 4.4
ML stacking ensemble models AGE for each time series type.

Time Series Type	Measure	*Voting Regressor*	*Stacking ML Ensemble*		
		Average of top 2	LR Ensemble	SVR Ensemble	GB Ensemble
Trend	AGE (RMSE+MAE)/2	*9.69*	*9.95*	*10.08*	*11.27*
Seasonal		*10.30*	*10.56*	*10.53*	*11.71*
Combined		*10.73*	*10.86*	*10.80*	*14.64*
Random		*8.91*	*8.96*	*8.94*	*10.48*

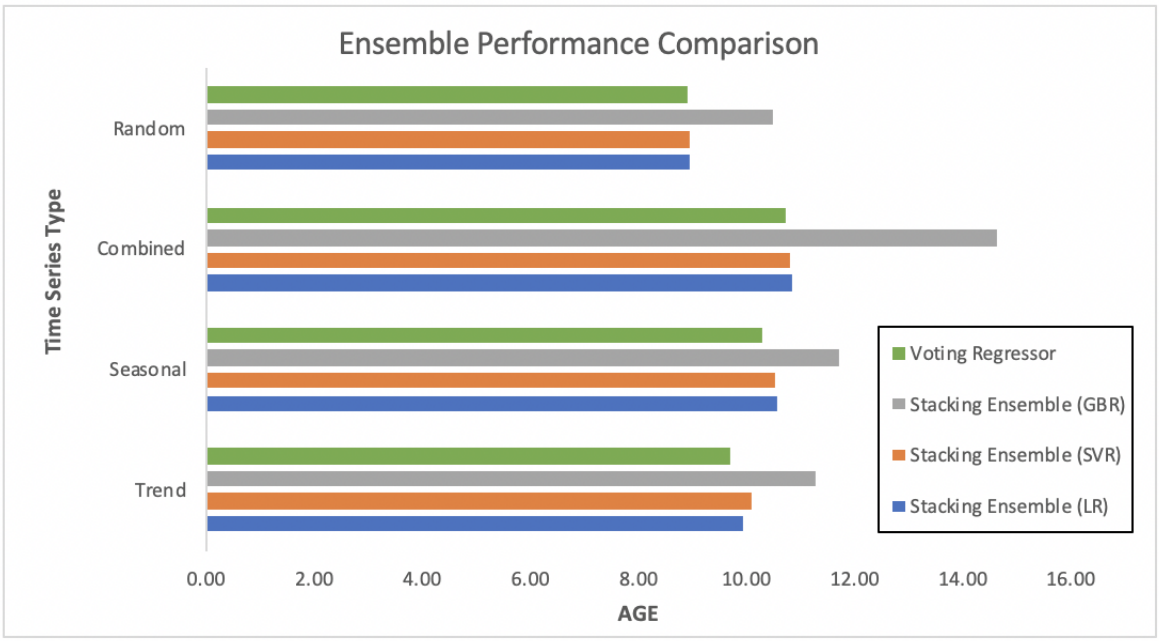

Figure 4.7. ML ensembles performance comparison with each of the four time series types.

4.3 Benchmarking Phase

The steps follow the logical order by first determining how far the imputation helps in prediction. In case it helps, the next step follows to compare it with the other models in the market to determine the effectiveness of the model against the other ones.

4.3.1 Impact of Imputation on Forecasting

The first step is to determine whether imputation has an impact on forecasting or not. In order to do so, different imputation models are compared against the strategy in which all the missing values are deleted (Complete Case Analysis - CCA). The utilized dataset in this experiment is California's weather data as well as strawberries yield and price information from Oxnard and Santa Maria stations downloaded from two publicly available websites [120, 122] from 2011 to 2019. The imputation models used are the following:

- *Last Observation Carried Forward (LOCF):* In this model the imputed value is based on the assumption that the last non-missing observed value for yield or price is carried forward. A bi-directional version is used in this work where preceding values are used to fill following missing values and vice versa.

- *Linear Function:* The linear interpolation function as used in [32] is a function in Python language which ignores the index and treats all the values as equally spaced. It sees the trend and then fills up the missing values using the trend in the dataset. In order to increase the robustness of the interpolation method the bi-directional interpolation is used which fills missing values in both directions (forward and backward). This enables filling any missing values in the first row of the dataset since those are not filled by the forward interpolation.

- *KNN-Imputation:* The KNN-imputation is a methodology where the k nearest neighbours are used to impute the missing values in the dataset. This way the cross-relation between variables in the multivariate time series is intact as the all the variables are considered while imputing the missing values.

- *LSTM:* In this LSTM the imputation framework is used to impute the missing values.

The imputation models are applied to the incomplete California time series. The output files imputed by each of these models, (weather, yield and price files), are used as input to the prediction model of simple LSTM. The imputation performance of each of the tested imputation models is judged the prediction accuracy.

To make the comparison, simple LSTM model is trained using the imputed weather, price and yield files to predict yield and prices 5 weeks ahead using the previous 5 months' weather. The winning imputation model is the one with the least AGM value as mentioned in Section 2.6.

In California dataset, yield and price data are imputed differently from the weather. The yield and price data are interpolated in two ways: Filling using closest time value based on the Last observed value (LOCF) and simple LSTM model. LOCF is based on filling the yield and price with the previous available value (one day before or after) and if that is not available, the same dates of the previous year for the same or closest station is used. Therefore, it is done by first looking one day behind, if the price from Santa Maria station is available the missing value is filled with that last observed price if not then the price one day ahead is used instead. If both preceding and following values are missing, the window is expanded to consider Santa Maria's yields and prices one year back. If Santa Maria's last year value for the same date is present then it is taken otherwise the value the

day before or after is taken based on availability. In situations where none of these values are present, the value is filled using the closest station, Oxnard, and the same methodology of selecting the date is reapplied. Weather, on the other hand, is imputed using LSTM in addition to the Linear function.

Four files are created based on four different combinations of these imputation methods, see Table 4.5. All the four files are created based on the five months weather lag and the 5 weeks ahead prediction. It should be noted that the M2M file represents the file preprocessed by the CCA; no imputation since all records with missing values are discarded.

Table 4.5
Imputation combinations.

Imputation Test Files	Interpolated Parameter	No Missing CCA (M)	Filling LOCF (FL)	Linear Function (LN)	LIME (LM)
M2M (No Imputation)	*Weather*	√			
	Yield & price	√			
M2LN	*Weather*			√	
	Yield & price	√			
FL2LM	*Weather*				√
	Yield & price		√		
LM2LM	*Weather*				√
	Yield & price				√

Price and yield predictions are then conducted using the three imputed files as input to the simple DL LSTM prediction model along with the file without any imputation, M2M. Five months weather lag is used for predicting the price 5 weeks ahead as illustrated in Table 4.6 .

Table 4.6
Dates mapping.

Price & Yield Date (YYYY-MM-DD)	Price ($/Pound)	Yield (Pounds/Acre)	Weather Date (YYYY-MM-DD)
2011-09-25	$ 0.52	110	2011-04-03
2011-09-26	$ 0.92	132	2011-04-04
2011-09-27	$ 0.92	122	2011-04-05

Based on the aggregated measure AGM, it is found that the LSTM imputation model for weather, price and yield leads to the best prediction performance across both applications of price and yield prediction using weather; W2P and W2Y. Furthermore, it can also be seen that with imputation there is an improvement in the prediction thereby making imputation a vital task in the pre-processing stage. The LM2LM input file results in the least AGM as listed in Table 4.8 and illustrated in Figure 4.9.

Table 4.7
Prediction results of different imputation techniques.

App.	Test Files	MAPE	MAE	R^2	RMSE	AGM
W2P	M2M	45.23	0.49	0.29	0.65	0.41
	M2LN	41.73	0.53	0.17	0.67	0.50
	FL2LM	41.83	0.45	0.29	0.60	0.37
	LM2LM	40.24	0.42	0.34	0.57	**0.33**
W2Y	FL2LM	677.64	45.90	0.74	71.95	15.30
	LM2LM	607.70	43.92	0.74	71.46	**14.85**

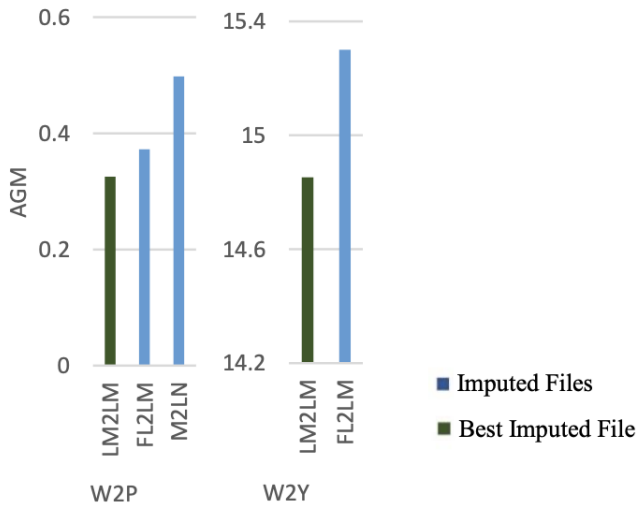

Figure 4.8. Prediction results of different imputation techniques.

4.3.2 Selection of Forecasting Model

Once the importance of imputation is deduced the next step is to validate the claim of the framework as mentioned in Section 4.2.2. In order to gauge the effectiveness of the imputation model, the imputed data is fed to the forecasting model and AGM is calculated.

To select the forecasting model, 9 DL models and 3 ensemble models are used. The models are as follows:

1. DL models
   (a) Encoder-Decorder
   (b) SeriesNet with LSTM with Attention
   (c) SeriesNet with GRU with Attention
   (d) SeriesNet with CNN-LSTM with Attention
   (e) SeriesNet with ConvLSTM with Attention
   (f) SeriesNet with CNN-LSTM ConvLSTM with Attention
   (g) SeriesNet with LSTM GRU with Attention
   (h) SeriesNet with CNN-LSTM GRU with Attention
   (i) SeriesNet with ConvLSTM GRU with Attention
2. Ensemble Models
   (a) Voting Regressor
   (b) Stacking ML Ensemble (SVR)
   (c) Stacking ML Ensemble (GBR)

The experimentation is carried out on daily prices for Brent oil. The prices are forecasted for 1, 2, 3 and 7 days ahead. The univariate time series is converted to a supervised learning problem with a sequence of 10 days as input and some point after that sequence as output depending on the step ahead being utilized. For each horizon, AGM is calculated and then overall AAGM is calculated.

The AAGM results are illustrated in Table 4.8 and in Figure 4.9. As evident from the results SeriesNet with GRU outperforms all the other models. The reason for the performance is by virtue of its architecture. The architecture has dilated CNN which caters

the long- term spatial features and GRU which cater the temporal features. Additionally, it is also observed that SeriesNet with GRU has a lower AAGM than SeriesNet with LSTM. This is due to the more straightforward structure of GRU, which allows the error to backpropagate without vanishing too quickly as it bypasses multiple temporal steps [25].

It is also observed that voting regressor gives a comparable result with SeriesNet with GRU but since voting regressor requires more computational time hence SeriesNet with GRU is a better choice.

The experiment aids in determining the forecasting model that is to be used while gauging the imputation against the market models. The results illustrates that SeriesNet with GRU is the optimal choice and hence it will be used for all the analysis on market benchmarking.

Table 4.8
Prediction results of different imputation models.

Models	Average MAE	Average MSE	Average R2	AAGM
SeriesNet with GRU	*1.308*	*3.071*	*0.942*	*0.105*
Voting Regressor	*1.319*	*3.082*	*0.942*	*0.105*
SeriesNet with LSTM	*1.342*	*3.129*	*0.941*	*0.106*
SeriesNet with CNN-LSTM & ConvLSTM	*1.317*	*3.130*	*0.941*	*0.108*
SeriesNet with GRU & LSTM	*1.315*	*3.148*	*0.941*	*0.109*
SeriesNet with ConvLSTM & GRU	*1.339*	*3.220*	*0.940*	*0.114*
SeriesNet with ConvLSTM	*1.344*	*3.244*	*0.939*	*0.114*
Stacking ML Ensemble (GBR)	*1.427*	*3.602*	*0.939*	*0.119*
SeriesNet with CNNLSTM	*1.450*	*3.775*	*0.929*	*0.142*
SeriesNet with CNNLSTM & GRU	*1.449*	*3.797*	*0.929*	*0.146*
Encoder-Decoder	*2.117*	*6.941*	*0.871*	*0.310*
Stacking ML Ensemble (SVR)	*1.901*	*8.353*	*0.860*	*0.346*

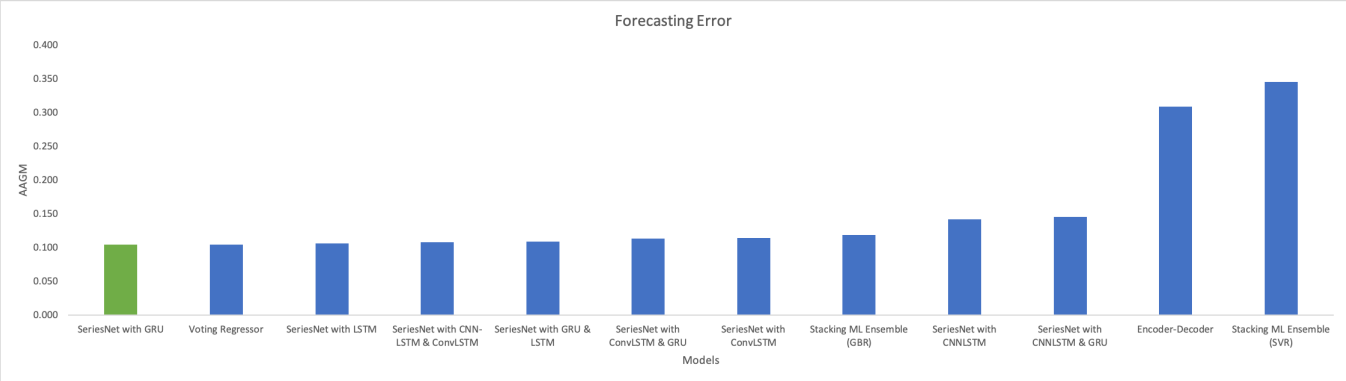

Figure 4.9. Prediction results of different imputation models.

4.3.3 Benchmarking on Oil Dataset

Now that the forecasting model is selected, the benchmarking is carried out. The road map followed for the benchmarking in case of oil dataset is as follows:

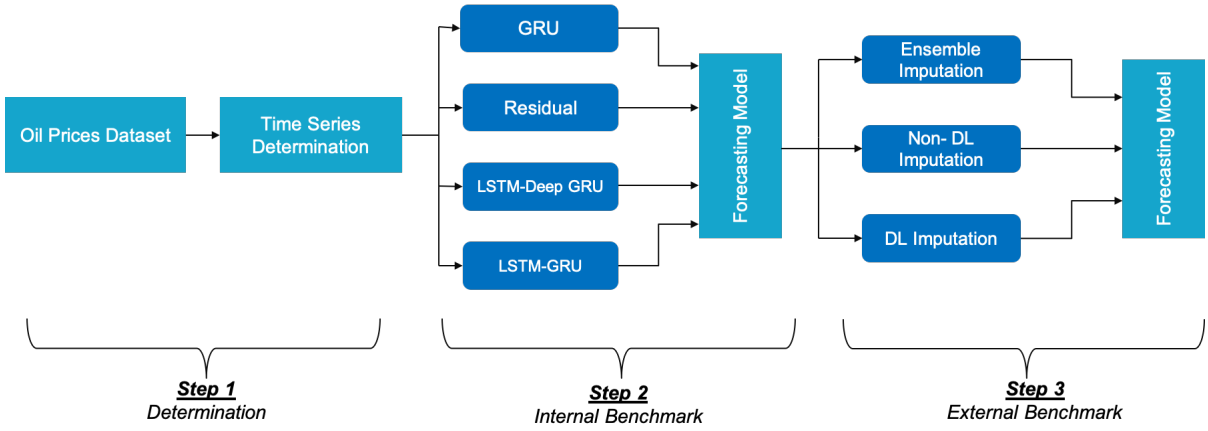

Figure 4.10. Roadmap for Oil dataset.

4.3.3.1 Identification of TS Type

As evident from the road map, the first step is the identification of the time series that the dataset is following. For the Identification, the time series is decomposed into trend and seasonal components and each of the components is then analyzed separately.

For the trend component, the data is divided into two sections and for each section the mean is calculated. In case the difference between the means of both sections is within 10% of each other, the data is classified as having no trend. Conversely, if the means of both sections have a difference greater than 10%, the data is said to have trend in it. The threshold of 10% is set after conducting various tests on different time series datasets and it is found to be most suitable. Thresholds below 10% are found to be too stringent whereas more than 10% are found to be quite relaxed.

For identifying the seasonal component, the Savitzky–Golay filter [108] is employed. The seasonal component is passed through the filter and MAE is calculated between the filtered data and the original data. If the MAE is found to be in the proximity of zero, the data is declared as seasonal otherwise in case it is greater than 1, it is concluded that there is no seasonality component in it.

On the basis of the above tests, the category is decided. The summarized version is illustrated in Table 4.9

Table 4.9
Time series classification combinations

Trend Test	Seasonality Test	Time Series Category
Passed	*Failed*	*Trend*
Failed	*Passed*	*Seasonal*
Passed	*Passed*	*Combined*
Failed	*Failed*	*Random*

The time series classification test is applied on the oil dataset and the results suggest that the oil dataset follows a combined time series that has components of both trend and seasonality. Under this circumstance, the most suitable imputation method is the ensemble of the GRU and Residual as per the results of Section 4.2.2.

4.3.3.2 Internal Benchmarking

In order to validate this claim, the dataset is imputed via the common top 4 imputation methods among the set of all top 2 models found internally by the system for all the 4 time series types. The hypothesis is that the top performing models are as per the framework recommended earlier. The results of the four imputation models are gauged using the forecasting model in which the forecasting is performed for the horizon of 1,2,3 and 7 days ahead. The metric used is the AAGM which is the average of the AGM of all the steps ahead (i.e. 4 in this case).

The results of the internal benchmarking are illustrated in Figure 4.11. It is observed that the top two performing models are GRU and Residual GRU which validates the claim stated earlier.

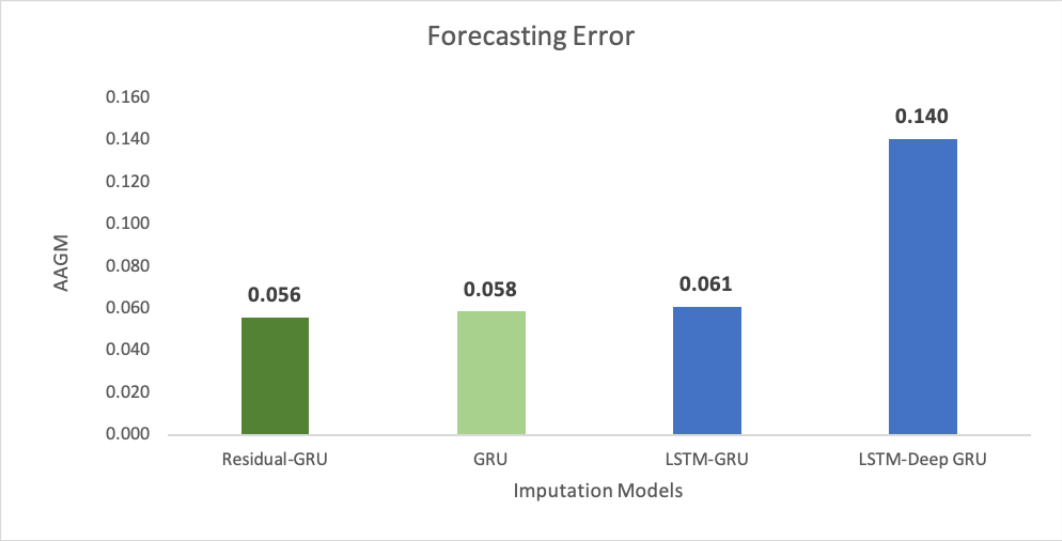

Figure 4.11. Prediction results of different imputation models.

4.3.3.3 External Benchmarking

Here the ensemble model is compared with the common market deep and non-deep models. The utilized deep learning model is LIME LSTM whereas Interpolation technique using linear function is used in the category of non-deep learning models. The ensemble is the average of the results of GRU and Reisdual-GRU. The horizon is the same as it is in internal

benchmarking. To gauge the performance AAGM is calculated which is the average of the AGM of all the steps ahead (i.e. 4 in this case).

The results of this step is illustrated in Figure 4.12. It is found via the results that the ensemble of GRU and Residual GRU is outperforming the common models of the market. As evident from the figure, when compared with the market deep learning model (LIME-LSTM), ensemble imputation provides an improvement of 0.5% in forecasting whereas, when compared with non-deep learning model (interpolation), the improvement is found to be around 2%.

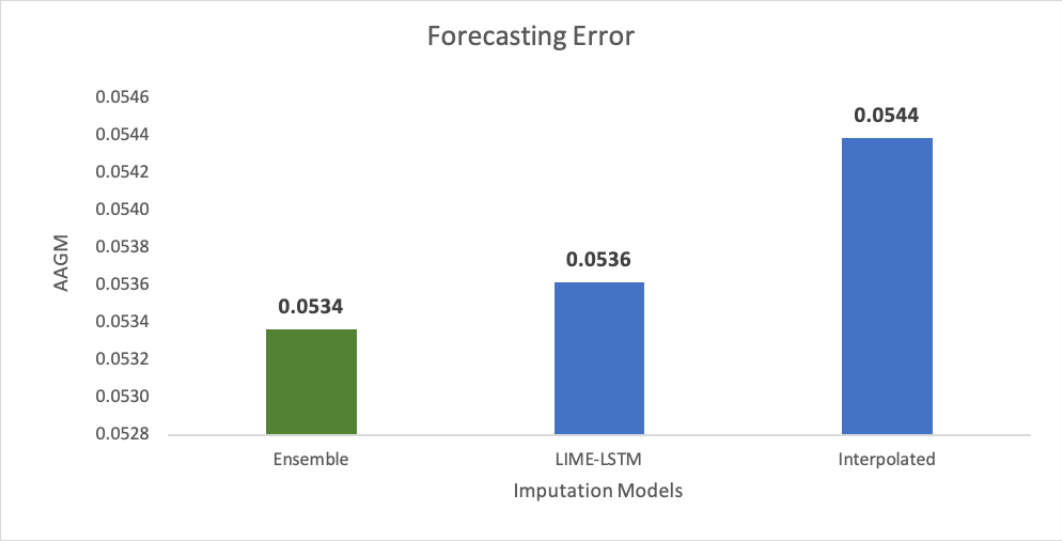

Figure 4.12. Prediction results of different imputation models.

Furthermore, in order to compare the models with naive forecasting, MASE is also calculated for each model. The results are illustrated in the Table 4.10. As evident from the Table 4.10, all the models have an MASE of less than one, which portrays that the models are better than naive forecasting.

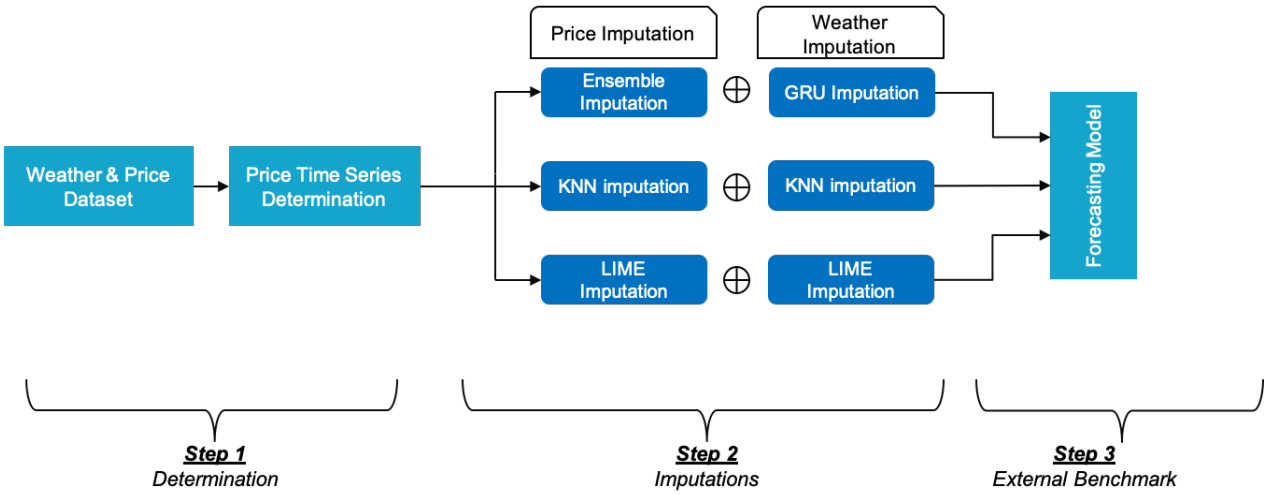

Figure 4.13. Roadmap for Weather and Price Dataset.

Table 4.10
MASE results of different imputation models.

Model	MASE
Ensemble	*0.525*
LIME-LSTM	*0.509*
Interpolated	*0.510*

4.3.4 W2P Benchmarking

For W2P the roadmap is illustrated in Figure. 4.13. As evident from the Figure 4.13 the steps are more or less the same as the oil dataset. Since the weather dataset is a multivariate time series, the time series determination test is not conducted on it. However, the test for time series is conducted on price dataset.

For weather and price different imputation models are employed. For ensemble imputation, weather is imputed via GRU because it is found in Section 4.2.2 that GRU based neural networks are found to be most suitable for imputation, though the complexity varies from time series to time series. Whereas for KNN and LIME, the same model is used for

both price and weather. After imputation the weather for 20 weeks prior to the price forecasting date is taken as input to forecast future prices 5 weeks from that date. This means that the input weather data is used to train the model to forecast the price 5 week after the weather period. To get the input variables, weather values of each day are stacked horizontally so that each day has 13 weather features, hence there are 1820 features for 140 days (20 weeks) which get mapped to one price at some step into the future. Principal component analysis (PCA) is applied to compress the 1820 features to 104 while retaining 77% of the variance. These daily weather values across 20 weeks affect the price. The horizontal stacking of the daily data is created and implemented to transform the daily data into a set covering a 20-week span. This stacking introduces a new problem since the number of features for the model became 1820. This number is too much for the size of the dataset and could lead to possible overfitting hence the need for PCA to reduce the dimensionality to 104. Depending on the model being applied, this input data with a dimensionality of 104 is reshaped again to suit the kind of input expected by the model.

For price, the dataset is passed through the time series classifier and it is found that the price follows a combined time series that is it has a component of both trend and seasonality. Based on the framework in Section 4.2.2, ensemble of GRU and Residual GRU imputation is conducted on the prices. The experiment doesn't require internal benchmarking as it is already conducted and validated in Section 4.3.3.

For external benchmarking, LIME LSTM and KNN-imputation are conducted to be compared against the ensemble imputation. The imputed files of weather and price are combined as per the time frames and fed into the forecasting model that is SeriesNet with GRU. To gauge the performance of the forecasting, AGM is calculated for each of the methods. Furthermore, based on validation results, hyper- parameters such as dropout is also tweaked to either tackle overfitting or underfitting. The best weights are saved during training based on the least validation loss so as to combat overfitting caused by too many epochs. The results are illustrated in Figure 4.14. As evident from the results, the ensemble model provides the best forecasting results. As evident from the figure, ensemble model provides an improvement of around 57% in forecasting errors when compared with deep learning model (LIME-LSTM) and an improvement of 66% when compared with non-deep learning model (KNN-imputation).

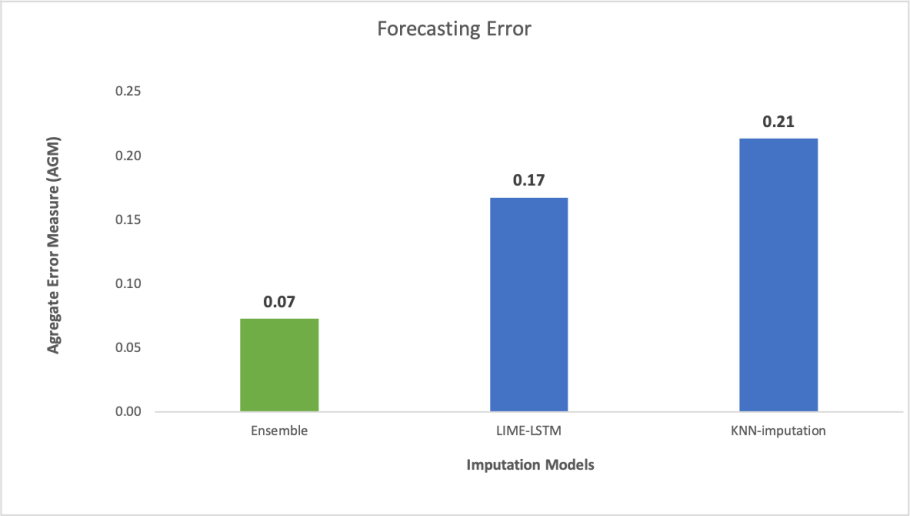

Figure 4.14. Forecasting results of different imputation models.

The models are also compared with naive forecasting, and hence MASE is also calculated for each model. The results are illustrated in the Table 4.11. As evident from the Table 4.11, all the models have an MASE of less than one, which portrays that the models are better than naive forecasting.

Table 4.11
MASE results of different imputation models.

Model	MASE
Ensemble	*0.37*
LIME-LSTM	*0.54*
KNN-Imputation	*0.55*

4.3.5 W2Y Benchmarking

For W2Y, the following roadmap is followed.

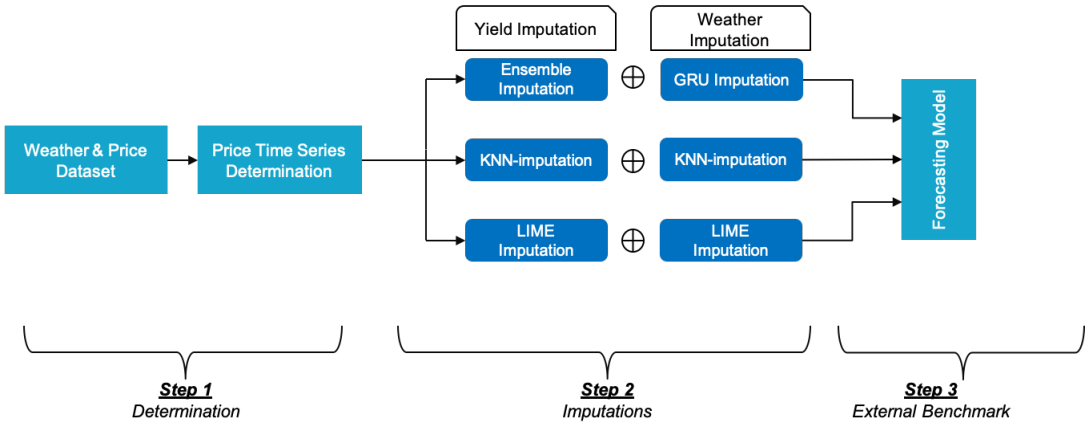

Figure 4.15. Roadmap for Weather and Yield Dataset.

As evident from the Figure 4.15 the steps are more or less the same as the experiment in Section 4.3.4. Since the weather dataset is a multivariate time series, the time series determination test is not conducted on it. However, the test for time series is conducted on the yield dataset.

For weather the same preprocessing is conducted on it as mentioned in Section 4.3.4. For yield, however, the dataset is passed through the time series classifier which shows that the yield follows a seasonal time series. Based on the framework in Section 4.2.2, the ensemble of Residual GRU and LSTM-Deep GRU imputation is conducted on the yield. The experiment doesn't require internal benchmarking as it is already conducted and validated in Section 4.3.3.

For external benchmarking, LIME-LSTM and KNN-imputation is conducted as the comparison against the ensemble imputation. The imputed files of weather and price are combined as per the time frames and fed into the forecasting model that is SeriesNet with GRU. To gauge the performance of the forecasting, AGM is calculated for each of the methods. Furthermore, based on validation results, hyper- parameters such as dropout are also tweaked to either tackle overfitting or underfitting. The best weights are saved during training based on the least validation loss so as to combat overfitting caused by too many epochs. The results are illustrated in Figure 4.16. As evident from the results, the ensemble model provides the best forecasting results. It is being observed from the figure that when compared with LIME, ensemble imputation model provides an improvement of 35% whereas, with KNN-imputation the improvement is found to be around 67%.

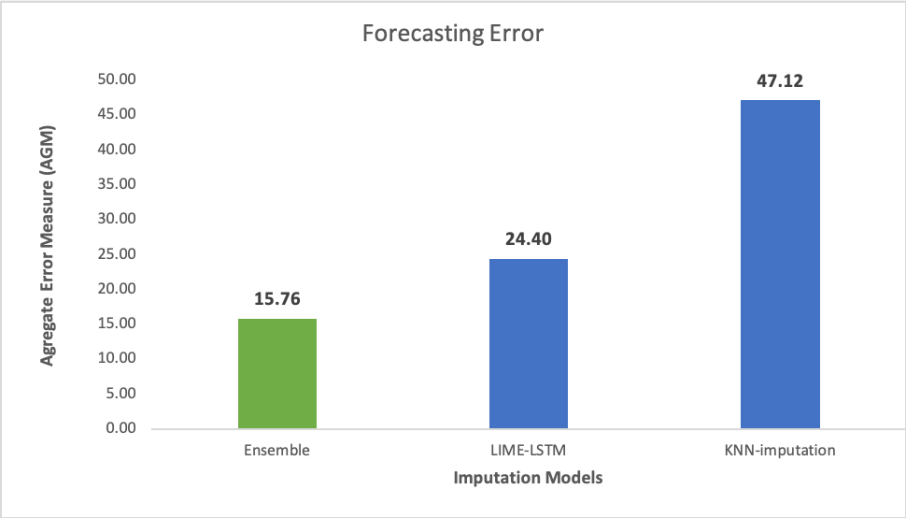

Figure 4.16. Forecasting results of different imputation models.

The models are also compared with naive forecasting by calculating MASE. The results of which are illustrated in the Table 4.12. As evident from the Table 4.12, all the models have an MASE of less than one, which portrays that the models are better than naive forecasting.

Table 4.12
MASE results of different imputation models.

Model	MASE
Ensemble	*0.40*
LIME-LSTM	*0.42*
KNN-Imputation	*0.51*

4.4 Conclusion

The chapter started off with the determination of the approach most fit for imputation. In order to do this, corn yield dataset is being utilized and it was found that the deep

learning approach is found to be optimal for imputation.

The next step in the chapter, is the determination of the best suited imputation model for each time series. The reason for this is because, every time series follow a different trajectory therefore, it is not possible to generalize one model to all the types of time series. The experiment results, further validated this claim and this enabled us to come up with a recommendation framework for each type of time series in order to have the best imputation results.The results of the models are further improved using different ensemble techniques making the models more robust and effective.

Once the models are determined, the next step is to benchmark them with the models used in literature and guage their effective with respective to them. To perform this, three different type of incomplete time series datasets are being used and the imputation is being performed with recommended models and literature models. In order to determine the effectiveness of imputation, forecasting model is being developed and used. It is found from the experimentation that the recommended model gives a lesser forecasting error as opposed to the other models. In oil price dataset, it gives an improvement of upto 2% whereas in W2P and W2Y it gives an improvement of upto 66%.

Chapter 5

Conclusion

The objective of this thesis is to tackle the problem of fresh produce yield and price forecasting. However, one problem which hinders this activity is the problem of missing values in the dataset. Therefore, this research handles this problem by introducing various novel imputation frameworks and models to provide greater accuracy and gauge its impact on FP forecasting. There are models which exist in the current literature but they are either limited to certain domains, e.g. medical fields, or assume linearity in the system which is usually not the case. This thesis fills the void by introducing various imputation models that range from simple to complex models including ensemble techniques. Furthermore, it also provides a recommendation framework for different types of time series which are commonly observed. The performance of these imputations are gauged by the improvement in forecasting for which multiple novel attention-based models are developed. Strawberries are selected as the case study for FP because of its extra difficulty caused by its short shelf life.

Since the research tackles two aspects of the problem, imputation and forecasting, both aspects are discussed and experimented and throughout the work, it is presumed that the time series is stationary. The results from the imputation phase, suggests that the averaging ensemble leads to having the least forecasting error and hence is the optimal solution for the missing values. It is also observed that the GRU based neural networks generally perform better as opposed to other models though the complexity of the model varies from time series to time series.

In the forecasting phase, amongst different novel forecasting, SeriesNet with GRU is found to outperform other models. Moreover, the imputation framework is being validated and compared with the existing models in the market and it is found that the imputation

of via the framework results in a better forecasting results when compared to other market models. Quantitative in the oil dataset, the ensemble model provided a 0.5% and 2% improvement in the forecasting error when compared with market deep and non-deep method respectively. In W2P dataset the improvement noted was 57% when compared with LIME and 66% when compared with KNN-imputation. In the W2Y, the improvement was 35% and 67% when compared with LIME and KNN-imputation.

Despite having a versatile framework for imputation, the model is limited to cater only missing values that occur at random which is more probable and frequently observed. Therefore, the future work can cater this aspect and alter the model to be robust enough to tackle large chunks of missing values. The reason that large chunk of missing values is difficult to impute is because each time series carries information within itself and the absence of a chunk of values makes it harder for the neural network to determine the trend of the series. Furthermore, advanced neural networks such as Generative Adversarial Networks (GANs) [53] can also be applied and explored for imputation.

In terms of forecasting, the following can be considered as the future tasks:

- Predicting FP farm-gate prices as a function of yield should be explored to determine the best approach to price modelling.
- Incorporate other external factors such as soil parameters, irrigation, etc. to improve model performance.
- Although California is the major supplier of strawberries, weather data from other areas which form the minority can be considered in determining strawberry prices.
- Satellite imagery which provide information vegetation and land can be explored as a source of additional information which deep learning models can utilize in modelling yield and price.
- Extending the work in this thesis to other FP similar to or different from strawberry and see how well it is generalizable to other crops.

References

[1] Corn & soybean prices 2008-2017 — kaggle. https://www.kaggle.com/ainslie/usda-wasde-monthly-corn-soybean-projections#USDAProj_Corn_2007to2008.csv. (Accessed on 12/20/2019).

[2] US Energy Information Administration. Spot prices for crude oil and petroleum products. https://www.eia.gov/dnav/pet/pet_pri_spt_s1_d.htm. (Accessed on 07/07/2020).

[3] Hamed Habibi Aghdam and Elnaz Jahani Heravi. Guide to convolutional neural networks. *New York, NY: Springer*, 10:978–973, 2017.

[4] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010.

[5] David LJ Alexander, Alexander Tropsha, and David A Winkler. Beware of r 2: simple, unambiguous assessment of the prediction accuracy of qsar and qspr models. *Journal of chemical information and modeling*, 55(7):1316–1322, 2015.

[6] Paul D Allison. *Missing data*, volume 136. Sage Publications, 2001.

[7] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[8] M. F. Anaghi and Y. Norouzi. A model for stock price forecasting based on arma systems. In *2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, pages 265–268, 2012.

[9] Aphex34. File:typical cnn.png - wikimedia commons. https://commons.wikimedia.org/wiki/File:Typical_cnn.png. (Accessed on 07/05/2020).

[10] Ibrahim Berkan Aydilek and Ahmet Arslan. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*, 233:25–35, 2013.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[12] Rita Beigaite, Tomas Krilavičius, and Ka Lok Man. Electricity price forecasting for nord pool data. In *2018 International Conference on Platform Technology and Service (PlatCon)*, pages 1–6. IEEE, 2018.

[13] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[14] Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

[15] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[16] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.

[17] Jean C Buzby, Jeanine T Bentley, Beth Padera, Cara Ammon, and Jennifer Campuzano. Estimated fresh produce shrink and food loss in us supermarkets. *Agriculture*, 5(3):626–648, 2015.

[18] RG Carpenter. Principles and procedures of statistics, with special reference to the biological sciences. *The Eugenics Review*, 52(3):172, 1960.

[19] NH Chan. Time series: Co-integration. 2001.

[20] Z. Chang, Y. Zhang, and W. Chen. Effective adam-optimized lstm neural network for electricity price forecasting. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 245–248, 2018.

[21] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.*, 8(1):6085, 2018.

[22] Sirapat Chiewchanwattana, Chidchanok Lursinsap, and Chee-Hung [Henry Chu]. Imputing incomplete time-series data based on varied-window similarity measure of data sequences. *Pattern Recognit Lett*, 28(9):1091 – 1103, 2007.

[23] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[24] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Proc. Mach. Learn. Healthcare Conf.*, pages 301–318, 2016.

[25] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[26] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

[27] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[28] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery.

[29] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[30] Jesús [Crespo Cuaresma], Jaroslava Hlouskova, Stephan Kossmeier, and Michael Obersteiner. Forecasting electricity spot-prices using linear univariate time-series models. *Applied Energy*, 77(1):87 – 106, 2004.

[31] Sandya De Alwis, Yishuo Zhang, Myung Na, and Gang Li. Duo attention with deep learning on tomato yield prediction and factor interpretation. In *Pacific Rim International Conference on Artificial Intelligence*, pages 704–715. Springer, 2019.

[32] Chaïm De Mulder, T Flameling, S Weijers, Youri Amerlinck, and Ingmar Nopens. An open software package for data reconciliation and gap filling in preparation of

water and resource recovery facility modeling. *Environmental Modelling & Software*, 107:186–198, 2018.

[33] R. Dey and F. M. Salem. Gate-variants of gated recurrent unit (GRU) neural networks. In *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, pages 1597–1600, 2017.

[34] A. Diallo, E. Kácsor, and M. Vancsa. Forecasting the spread between hupx and eex dam prices the case of hungarian and german wholesale electricity prices. In *2018 15th International Conference on the European Energy Market (EEM)*, pages 1–5, 2018.

[35] Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

[36] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[37] Philip Hans Franses. A note on the mean absolute scaled error. *International Journal of Forecasting*, 32(1):20–22, 2016.

[38] David A Freedman. *Statistical models: theory and practice*. Cambridge University Press, 2009.

[39] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[40] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367 – 378, 2002. Nonlinear Methods and Data Mining.

[41] R. Fu, Z. Zhang, and L. Li. Using LSTM and GRU neural network methods for traffic flow prediction. In *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Automat. (YAC)*, pages 324–328, 2016.

[42] David S Fung. Methods for the estimation of missing values in time series. Master's thesis, Edith Cowan University Perth, 2006.

[43] Min Gan, Yu Cheng, Kai Liu, and Gang lin Zhang. Seasonal and trend time series forecasting based on a quasi-linear autoregressive model. *Appl. Soft Comput.*, 24:13 – 18, 2014.

[44] Y. Gandge and Sandhya. A study on various data mining techniques for crop yield prediction. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 420–423, 2017.

[45] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

[46] Everette S Gardner Jr. Exponential smoothing: The state of the art—part ii. *International journal of forecasting*, 22(4):637–666, 2006.

[47] Anshul Garg and Bindu Garg. A robust and novel regression based fuzzy time series algorithm for prediction of rice yield. In *2017 International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 48–54. IEEE, 2017.

[48] C. Ge. A lstm and graph cnn combined network for community house price forecasting. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 393–394, 2019.

[49] N. N. Ghosalkar and S. N. Dhage. Real estate value prediction using linear regression. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–5, 2018.

[50] Eric Ghysels and Denise R Osborn. *The econometric analysis of seasonal time series*. Cambridge University Press, 2001.

[51] SA Glantz and BK Slinker. Primer of applied regression and analysis of variance, 2001. *Columbus, OH: McGraw-Hill Education*, 2.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[54] E. Hadavandi, H. Shavandi, and A. Ghanbari. A genetic fuzzy expert system for stock price forecasting. In *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, volume 1, pages 41–44, 2010.

[55] Andrew C Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[56] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[57] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998.

[58] Ming Hua and Jian Pei. Cleaning disguised missing data: A heuristic approach. In *KDD*, KDD '07, page 950–958, New York, NY, USA, 2007.

[59] Darren Hudson. Agricultural markets and prices. Technical report, 2007.

[60] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[61] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[62] J. Jagwani, M. Gupta, H. Sachdeva, and A. Singhal. Stock price forecasting using data from yahoo finance and analysing seasonal and nonseasonal trend. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 462–467, 2018.

[63] L. Jiang and G. Hu. Day-ahead price forecasting for electricity market using long-short term memory recurrent neural network. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 949–954, 2018.

[64] Z. Jiang and G. Shen. Prediction of house price based on the back propagation neural network in the keras deep learning framework. In *2019 6th International Conference on Systems and Informatics (ICSAI)*, pages 1408–1412, 2019.

[65] W.L. Junger and A. [Ponce de Leon]. Imputation of missing data in time series for air pollutants. *Atmos. Environ.*, 102:96 – 104, 2015.

[66] Terry L Kastens, Rodney Jones, and Ted C Schroeder. Futures-based price forecasts for agricultural producers and businesses. *Journal of Agricultural and Resource Economics*, pages 294–307, 1998.

[67] Saeed Khaki, Lizhi Wang, and Sotirios V Archontoulis. A cnn-rnn framework for crop yield prediction. *Frontiers in Plant Science*, 10:1750, 2020.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[70] Tarald O. Kvålseth. Cautionary note about r 2. *The American Statistician*, 39(4):279–285, 1985.

[71] Denis Kwiatkowski, Peter CB Phillips, Peter Schmidt, Yongcheol Shin, et al. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of econometrics*, 54(1-3):159–178, 1992.

[72] Y. Li, R. Liang, Z. Li, J. Gao, Y. Wang, and T. Wu. Research on electricity price forecasting method based on genetic algorithm and neural network in power market. In *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–6, 2018.

[73] Hong Lin, Yunzi Hua, Leiming Ma, and Lei Chen. Application of convlstm network in numerical temperature prediction interpretation. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, ICMLC '19, page 109–113, New York, NY, USA, 2019. Association for Computing Machinery.

[74] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[75] Zachary C Lipton, David Kale, and Randall Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Proc. Mach. Learn. Healthcare Conf.*, pages 253–270, 2016.

[76] H. Liu and B. Song. Stock price trend prediction model based on deep residual network and stock price graph. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 02, pages 328–331, 2018.

[77] Lon-Mu Liu, Siddhartha Bhattacharyya, Stanley L Sclove, Rong Chen, and William J Lattyak. Data mining on time series: an illustration using fast-food restaurant franchise data. *Computational Statistics & Data Analysis*, 37(4):455–476, 2001.

[78] S. Lu, Z. Li, Z. Qin, X. Yang, and R. S. M. Goh. A hybrid regression technique for house prices prediction. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 319–323, 2017.

[79] Q. Ma, S. Li, L. Shen, J. Wang, J. Wei, Z. Yu, and G. W. Cottrell. End-to-end incomplete time-series modeling from linear memory of latent variables. *IEEE Transactions on Cybernetics*, pages 1–13, 2019.

[80] C. R. Madhuri, G. Anuradha, and M. V. Pujitha. House price prediction using regression techniques: A comparative study. In *2019 International Conference on Smart Structures and Systems (ICSSS)*, pages 1–5, 2019.

[81] R. Medar, V. S. Rajpurohit, and S. Shweta. Crop yield prediction using machine learning techniques. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–5, 2019.

[82] Meeradevi and H. Salpekar. Design and implementation of mobile application for crop yield prediction using machine learning. In *2019 Global Conference for Advancement in Technology (GCAT)*, pages 1–6, 2019.

[83] Lucie Michel and David Makowski. Comparison of statistical models for analyzing wheat yield time series. *PLoS One*, 8(10):e78615, 2013.

[84] Sparsh Mittal. A survey of fpga-based accelerators for convolutional neural networks. *Neural computing and applications*, pages 1–31, 2020.

[85] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. Comparison of different methods for univariate time series imputation in R, 2015.

[86] Shinichi Nakagawa and Robert P. Freckleton. Missing inaction: the dangers of ignoring missing data. *Trends Ecol. Evol.*, 23(11):592 – 596, 2008.

[87] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Front. Neurorobot.*, 7:21, 2013.

[88] Petteri Nevavuori, Nathaniel Narra, and Tarmo Lipping. Crop yield prediction with deep convolutional neural networks. *Computers and electronics in agriculture*, 163:104859, 2019.

[89] A. Nigam, S. Garg, A. Agrawal, and P. Agrawal. Crop yield prediction using machine learning algorithms. In *2019 Fifth International Conference on Image Information Processing (ICIIP)*, pages 125–130, 2019.

[90] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[91] AN Patowary, PC Bhuyan, MP Dutta, J Hazarika, PJ Hazarika, et al. Development of a time series model to forecast wheat production in india. *Environment & Ecology*, 35(4D):3313–3318, 2017.

[92] S Madeh Piryonesi and Tamer E El-Diraby. Role of data analytics in infrastructure asset management: Overcoming data size and quality problems. *Journal of Transportation Engineering, Part B: Pavements*, 146(2):04020022, 2020.

[93] Kim Plunkett and Jeffrey L Elman. *Exercises in rethinking innateness: A handbook for connectionist simulations*. Mit Press, 1997.

[94] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[95] Robi Polikar. Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer, 2012.

[96] P. Przymus, Y. Hmamouche, A. Casali, and L. Lakhal. Improving multivariate time series forecasting with random walks with restarts on causality graphs. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 924–931, 2017.

[97] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

[98] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[99] Akhter Mohiuddin Rather, Arun Agarwal, and V.N. Sastry. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Syst. Appl.*, 42(6):3234 – 3241, 2015.

[100] Mrinmoy Ray, Anil Rai, V Ramasubramanian, and KN Singh. Arima-wnn hybrid model for forecasting wheat yield time-series data. *J. Ind. Soc. Agric. Stat*, 70(1):63–70, 2016.

[101] P Chandra Shaker Reddy and A Sureshbabu. An applied time series forecasting model for yield prediction of agricultural crop. In *International Conference on Soft Computing and Signal Processing*, pages 177–187. Springer, 2019.

[102] Alvin C Rencher. *Methods of multivariate analysis*, volume 492. John Wiley & Sons, 2003.

[103] L. Rokach and O. Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

[104] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33(1-2):1–39, 2010.

[105] Lior Rokach and Oded Z Maimon. *Data mining with decision trees: theory and applications*, volume 69. World scientific, 2008.

[106] K. B. Sahay and K. Singh. Short-term price forecasting by using ann algorithms. In *2018 International Electrical Engineering Congress (iEECON)*, pages 1–4, 2018.

[107] L. Sayavong, Z. Wu, and S. Chalita. Research on stock price prediction method based on convolutional neural network. In *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, pages 173–176, 2019.

[108] R. W. Schafer. What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine*, 28(4):111–117, 2011.

[109] Gabriel L Schlomer, Sheri Bauman, and Noel A Card. Best practices for missing data management in counseling psychology. *J. Couns. Psychol.*, 57(1):1, 2010.

[110] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[111] Hilary L Seal. *The historical development of the Gauss linear model*. Yale University, 1968.

[112] A Shabri, R Samsudin, Z Ismail, et al. Forecasting of the rice yields time series forecasting using artificial neural network and statistical model. *Journal of Applied Sciences*, 9(23):4168–4173, 2009.

[113] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[114] Z. Shen, Y. Zhang, J. Lu, J. Xu, and G. Xiao. Seriesnet:a generative time series forecasting model. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[115] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wangchun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 802–810. Curran Associates, Inc., 2015.

[116] A. Shiri, M. Afshar, A. Rahimi-Kian, and B. Maham. Electricity price forecasting using support vector machines by considering oil and natural gas price impacts. In *2015 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pages 1–5, 2015.

[117] Manish Shukla and Sanjay Jharkharia. Agri-fresh produce supply chain management: a state-of-the-art literature review. *International Journal of Operations & Production Management*, 2013.

[118] K. Silverstein. Trump administration to divert more of california's water to farming, impacting power production and wildlife. https://www.forbes.com/sites/kensilverstein/2020/02/21/trump-administration-to-divert-more-of-californias-water-to-farming-impacting-pow#521d69672195. (Accessed on 07/04/2020).

[119] James H Stock. Forecasting economic time series. *A Companion to Theoretical Econometrics, Blackwell Publishers*, pages 562–84, 2001.

[120] The California strawberry Commission website. Home — california strawberry commission. https://www.calstrawberry.com/en-us/. (Accessed on 07/07/2020).

[121] K. Sutiene, G. Vilutis, and D. Sandonavicius. Forecasting of GRID job waiting time from imputed time series. *Elektron Elektrotech*, 114(8):101–106, Oct. 2011.

[122] The California Irrigation Management Information System. Cimis. https://cimis.water.ca.gov/. (Accessed on 07/07/2020).

[123] A. S. Terliksiz and D. T. Altýlar. Use of deep neural networks for crop yield prediction: A case study of soybean yield in lauderdale county, alabama, usa. In *2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, pages 1–4, 2019.

[124] U Thissen, R Van Brakel, AP De Weijer, WJ Melssen, and LMC Buydens. Using support vector machines for time series prediction. *Chemometrics and intelligent laboratory systems*, 69(1-2):35–49, 2003.

[125] Andres M. Ticlavilca and Dillon M. Feuz. Forecasting agricultural commodity prices using multivariate bayesian machine. 2010.

[126] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays . *Bioinformatics*, 17(6):520–525, 06 2001.

[127] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, volume 01, pages 7–12, 2017.

[128] Dalhousie University and University of Guelph. Canada_food_price_report_eng_2018_.pdf. https://cdn.dal.ca/content/dam/dalhousie/pdf/management/News/News%20&%20Events/Canada_Food_Price_Report_Eng_2018_.pdf. (Accessed on 07/04/2020).

[129] H. Ur-Rehman, S. Mujeeb, and N. Javaid. Dcnn and lda-rf-rfe based short-term electricity load and price forecasting. In *2019 International Conference on Frontiers of Information Technology (FIT)*, pages 71–715, 2019.

[130] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013.

[131] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[132] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[133] F. Wang, Y. Zou, H. Zhang, and H. Shi. House price prediction approach based on deep learning and arima model. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 303–307, 2019.

[134] J. Wang and Y. Hu. An improved enhancement algorithm based on cnn applicable for weak contrast images. *IEEE Access*, 8:8459–8476, 2020.

[135] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 879–888. Curran Associates, Inc., 2017.

[136] P Werbos. Backpropagation through time: What it is and how to do it. In *IEEE Proceedings*, 1989.

[137] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[138] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[139] J. Wu and C. Lu. Computational intelligence approaches for stock price forecasting. In *2012 International Symposium on Computer, Consumer and Control*, pages 52–55, 2012.

[140] Y. Wu, L. Liu, J. Bae, K. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang. Demystifying learning rate policies for high accuracy training of deep neural networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1971–1980, 2019.

[141] A. Xavier. An introduction to convlstm - neuronio - medium. https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7. (Accessed on 07/05/2020).

[142] L. Xiao and T. Yan. Prediction of house price based on rbf neural network algorithms of principal component analysis. In *2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pages 315–319, 2019.

[143] Yi Xu, Longwen Gao, Kai Tian, Shuigeng Zhou, and Huyang Sun. Non-local convlstm for video compression artifact reduction. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[144] Zhe Xu and Yi Lv. Att-convlstm: Pm2.5 prediction model and application. In Yong Liu, Lipo Wang, Liang Zhao, and Zhengtao Yu, editors, *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 30–40, Cham, 2020. Springer International Publishing.

[145] Kouichi Yamaguchi, Kenji Sakamoto, Toshio Akabane, and Yoshiji Fujimoto. A neural network for speaker-independent isolated word recognition. In *First International Conference on Spoken Language Processing*, 1990.

[146] T. Ye. Stock forecasting method based on wavelet analysis and arima-svr model. In *2017 3rd International Conference on Information Management (ICIM)*, pages 102–106, 2017.

[147] Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. Generating chinese classical poems with rnn encoder-decoder. In Maosong Sun, Xiaojie Wang, Baobao Chang, and Deyi Xiong, editors, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 211–223, Cham, 2017. Springer International Publishing.

[148] Jinsung Yoon, William R. Zame, and Mihaela van der Schaar. Multi-directional recurrent neural networks : A novel method for estimating missing data. 2017.

[149] Ceylan Yozgatligil, Sipan Aslan, Cem Iyigun, and Inci Batmaz. Comparison of missing value imputation methods in time series: the case of turkish meteorological data. *Theor. Appl. Climatol.*, 112(1-2):143–167, 2013.

[150] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2015.

[151] M. Yu and J. Wu. Ceam: A novel approach using cycle embeddings with attention mechanism for stock price prediction. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–4, 2019.

[152] T. Yu, H. Peng, and W. Sun. Incorporating nonlinear relationships in microarray missing value imputation. *EEE/ACM Trans. Comput. Biol. Bioinform.*, 8(3):723–731, 2011.

[153] G.Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *Eur. J. Oper. Res.*, 160(2):501 – 514, 2005. Decision Support Systems in the Internet Age.

[154] C. Zheng and J. Zhu. Research on stock price forecast based on gray relational analysis and armax model. In *2017 International Conference on Grey Systems and Intelligent Services (GSIS)*, pages 145–148, 2017.

[155] S. Zhou, L. Zhou, M. Mao, H. Tai, and Y. Wan. An optimized heterogeneous structure lstm network for electricity price forecasting. *IEEE Access*, 7:108161–108173, 2019.

[156] Xiao-Hua Zhou, George J. Eckert, and William M. Tierney. Multiple imputation in public health research. *Stat Med*, 20(9-10):1541–1549, 2001.

[157] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artif. Intell.*, 137(1):239 – 263, 2002.