Major Project Report

On

A Study on Machine Learning-based Solar Energy Forecasting

Submitted by

191EE121 Hariharan Ayappane
191EE212 Harshal Dhake
191EE136 Naveen Venkat
191EE247 Sharvani Somayaji

Under the Guidance of

Prof. Dr. Yashwant Kashyap

Dept. of Electrical and Electronics Engineering,
NITK, Surathkal

Date of Submission: April 20, 2023

Department of Electrical and Electronics Engineering
National Institute of Technology Karnataka, Surathkal.
2023-2024

Declaration

We hereby declare that the project work entitled "A Study on Machine Learning-based Solar Energy Forecasting" submitted to the National Institute of Technology- Karnataka, is a record of an original work done by us under the guidance of Dr. Yashwant Kashyap, Assistant Professor Dept. of Electrical and Electronics Engineering. This project is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electrical and Electronics Engineering. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma

Certificate

This is to certify that the Major Project Report entitled

"A Study on Machine Learning-based Solar Energy Forecasting"

is submitted by
Hariharan Ayappane (191EE121)
Harshal Dhake (191EE212)
Naveen Venkat (191EE136)
Sharvani Somayaji (191EE247)

as a record of the work done by them is accepted as the Major Project submission in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electrical and Electronics Engineering, National Institute of Technology Karnataka, Surathkal

Supervisor Name:	Dr. Yashwant Kashyap
Designation:	Assistant Professor
Department:	Electrical and Electronics Engineering
University Name:	National Institute of Technology Karnataka, Surathkal

Project Guide

Abstract

This project studies the most frequently used ML-based techniques, primarily, Weather Research and Forecasting (WRF), Extreme Learning Machines (ELMs), Ensembling and Temporal Fusion Transformers. Each of these technqiues were applied to real-life solar energy (Global Horizontal Irradiance, GHI) time series data and forecasts were obtained for the range 0 minutes (immediate) to 90 minutes ahead. The first approach using WRF can be used for solar forecasting by incorporating solar radiation calculations into the modeling process, enabling accurate prediction of solar energy output. It is clear that the learning speed of feedforward neural networks is in general far slower than required and it has been a major bottleneck in their applications for past decades. Next we use a new learning algorithm called extreme learning machine (ELM) for single-hidden layer feedforward neural networks (SLFNs). After that, we explore an ensemble models to achieve more robust and accurate predictions. The model consists of Random Forest Regression, Deep Learning, LSTM and Ridge Regression as base models. It will be abbreviated to Ensemble RDLR (RF-DNN-LSTM-RR). Each base model is fine-tuned to promote high diversity and low correlation. By combining the stability and generalization ability of the RF-DNN with the memory retention capacity of the LSTM network, the proposed ensemble model is expected to provide an ideal predictor for time series forecasting of a stochastic process like weather. Finally we explore the application of SOTA models such as Temporal Fusion Transformer (TFT) on the time series data, to examine the efficacy of transformer based methods for energy forecasting. The model provides better interpretability while maintaining the accuracy. All of the techniques and models were implemented using Python and the results were obtained in terms of R^2 Score (Coefficient of Determination) for ease of comparison.

Contents

List of Figures

1 Introduction

Given the global sentiment shifting to exploration of renewable energy resources, it is important to understand the nature of these non-conventional resources, before tapping energy out of them at an economic scale. Solar Energy is, by far, the biggest renewable energy source, which also possesses high potential for mass energy generation. But similar to other renewable energy resources, solar energy is highly variable and erratic, making it difficult to efficiently forecast and hence exploit at an economic-scale. It is estimated that the energy consumption of human beings is around 0.3% of the gross photosynthetic energy produced by the Earth [2] and the rate of consumption is only growing. In order to keep up with this enormous growth and avoid catastrophic energy shortages we need to improve our renewable energy forecasting, production and distribution capabilities. Hence we have explored solar energy forecasting with focus on parameters like temperature, time of the year, humidity and location using various ML based techniques.

Physical Methods have been developed to understand the underlying patterns of the sun's movement and other atmospheric factors affecting the total energy throughput and making localised predictions. Physical Methods are based on the lower atmospheric or Numerical Weather Prediction (NWP) using parameters like temperature, pressure, surface roughness and obstacles as inputs. Physical methods are to increase the real resolution of the NWP model to achieve accurate prediction of the weather, which can be used reliably to forecast the Solar Energy in the particular region of study. Numerical weather prediction is based on the fact that the gases of the atmosphere obey a certain set of known physical principles and these physical laws can be used to predict the future state of the atmosphere from the current conditions. Hence, a large number of variables must be included and manipulated when considering the dynamic atmosphere and such operations require complex calculations. Due to this complex structure and method, physical methods require high computational power and have sluggish performance. Physical Methods are also often unable to forecast time series, like Solar Energy, which has irregular trends and swings in seasonalities and hence has lower accuracy in the long run. The Weather Research and Forecasting (WRF) Model is a state-of-theart mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications. It features two dynamic cores: a Data Assimilation system, and a software architecture supporting parallel computation and system extensibility. The data produced by the WRF contains a vast amount of atmospheric information in the form of numerous weather parameters, which can be used for Solar energy time series forecasting.

The WRF model is a fully compressible, nonhydrostatic model (with a hydrostatic option). Its vertical coordinate is a terrain-following hydrostatic pressure coordinate. The grid staggering is the Arakawa C-grid. The model uses the Runge-Kutta 2nd and 3rd order time integration schemes, and 2nd to 6th order advection schemes in both horizontal and vertical directions. It uses a time-split small step for acoustic and gravity-wave modes. The dynamics conserves scalar variables. The WRF model code contains ideal.exe and real.exe initialization programs, a numerical integration program (wrf.exe), and a program to do one-way nesting (ndown.exe). The WRF model Version supports a variety of capabilities. These include:

- Real-data and idealized simulations
- Various lateral boundary condition options for real-data and idealized simulations.
- Full physics options
- Non-hydrostatic and hydrostatic (runtime option)
- One-way, two-way nesting and moving nest
- Three-dimensional analysis nudging
- Observation nudging
- Applications ranging from meters to thousands of kilometers

Feedforward neural networks have been extensively used in many fields due to their ability: (1) to approximate complex nonlinear mappings directly from the input samples; and (2) to provide models for a large class of natural and artificial phenomena that are difficult to handle using classical parametric techniques. On the other hand, there lack faster learning algorithms for neural networks. The traditional learning algorithms are usually far slower than required. It is not surprising to see that it may take several hours, several days, and even more time to train neural networks by using traditional

methods. Traditionally, all the parameters of the feedforward networks need to be tuned and thus there exists the dependency between different layers of parameters (weights and biases). For past decades, gradient descent-based methods have mainly been used in various learning algorithms of feedforward neural networks. However, it is clear that gradient descent-based learning methods are generally very slow due to improper learning steps or may easily converge to local minima. And many iterative learning steps may be required by such learning algorithms in order to obtain better learning performance. SLFNs can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. After the input weights and the hidden layer biases are chosen randomly, SLFNs can be simply considered as a linear system and the output weights (linking the hidden layer to the output layer) of SLFNs can be analytically determined through simple generalized inverse operation of the hidden layer output matrices. Based on this concept, a simple learning algorithm is used for SLFNs called extreme learning machine (ELM) whose learning speed can be thousands of times faster than traditional feedforward network learning algorithms like back-propagation (BP) algorithm while obtaining better generalization performance. Different from traditional learning algorithms the proposed learning algorithm not only tends to reach the smallest training error but also the smallest norm of weights.

Recently, extreme learning machine (ELM) has been proposed for training single hidden layer feedforward neural networks (SLFNs). In ELM, the hidden nodes are randomly initiated and then fixed without iteratively tuning. Actually, the hidden nodes in ELM are even not required to be neuron alike. The only free parameters need to be learned are the connections (or weights) between the hidden layer and the output layer. In this way, ELM is formulated as a linear-in-the-parameter model which boils down to solving a linear system. Compared to traditional FNN learning methods, ELM is remarkably efficient and tends to reach a global optimum. Theoretical studies have shown that even with randomly generated hidden nodes, ELM maintains the universal approximation capability of SLFNs. With commonly used activation functions, ELM can attain the almost optimal generalization bound of traditional FNN in which all the parameters are learned. The advantages of ELM in efficiency and generalization performance over traditional FNN algorithms have been demonstrated on a wide range of problems from different fields. It is worth noting that ELM is generally much more efficient than support vector machines (SVMs), least square support vector machines (LS-SVMs) and other state-of-the-art algorithms. Empirical studies have shown that the generalization ability of ELM is comparable or even better than that of SVMs and its variants.

Machine Learning (ML) and Neural Networks (NN) are often used for time series forecasting. ML and NN models are networks made up of single cells which process the data passing through them. Recurrent Neural Networks (RNNs) are used for time series forecasting but they face the issue of long-term memory retention. Long sequences often require long-term memory retention for an accurate forecast, hence, Long Short Term Memory Networks (LSTM Networks) are used to overcome this problem. The LSTM network can be configured and varied using numerous parameters which change the overall structure of the network. This change in the structure of the network is accompanied by a drastic change in the forecasting accuracy of the model. Hence, the optimal network must be generated before using it for forecasting the time series data.

Another prevalent technique in forecasting involves combining or ensembling multiple base ML models in a logical manner to forecast time series data. Such a structure allows the model to learn the varying trends and seasonalities present in the data, as each sub-model present in the ensemble model is sensitive to different aspects of the data. Hence, this helps in maximising the understanding of patterns in the data. The ensemble technique used in this paper will be refered to as Ensemble RDLR (RF-DNN-LSTM-RR). 3.4

In a variety of time series forecasting problems, transformer models have also been used and have demonstrated cutting-edge performance. Parallelism in transformers has a major advantage over LSTM Networks and RNNs. They can take advantage of GPUs for parallelism (eg: BERT) because of the non-sequential processing of input data. Transformers are likely to be the most successful technique for extracting semantic correlations among items in a long sequence. However, in solar energy time series modelling, we must extract the temporal relationships from an ordered set of continuous points. But the self-attention mechanism leads to temporal loss, hence Temporal Fusion Transformers (TFTs) leverage self-attention to capture complex temporal dynamics of multiple time sequences by integrating the LSTM layers and attention heads in transformers.

In this paper, we study three ML based techniques described above and compare them with actual forecast data for a 90-minute ahead forecast. The paper makes the following contributions:

- The study and implementation of Extreme Learning Machines, Ensemble RDLR and Temporal Fusion Transformers in the context of time series forecasting and applying them to forecast solar

energy loads.

- Comparing the above methods for various time horizons of the forecast and derive the suitability and applicability of each of the models.

The remainder of this paper is structured into sections: Section 3 presents the methodology used to study, develop and implement the techniques for forecasting actual solar energy. Section 4 explores the results obtained from each of the techniques. The conclusion is presented in Section 5.

2 Literature Review

2.1 Extreme Learning Machines

ELMs have been explored extensively in the ML domain for over several years. Several works have been conducted to test, evaluated, modify and adopt ELM models for different machine learning problems. However, for this project we consider peculiar works which dealt with time series forecasting problems. Particularly, the following literature helped us to understand ELMs and implement them for time series forecasting.

Originally, Huang et al., 2006 [3] proposed the ELM algorithm for single-hidden layer feedforward neural networks (SLFNs). The learning speed of ELM was found to be extremely fast. The learning phase of ELM was completed in seconds or less than seconds for many applications. Previously, it seemed that there existed a virtual speed barrier which most classic learning algorithms couldn't break through and took very long time to train a feedforward network using classic learning algorithms even for simple applications. The traditional classic gradient-based learning algorithms may face several issues like local minima, improper learning rate and overfitting, etc. In order to avoid these issues, some methods such as weight decay and early stopping methods may need to be used often in these classical learning algorithms. The ELM tends to reach the solutions straightforward without such trivial issues. The ELM learning algorithm is simpler than most learning algorithms for feedforward neural networks.

Wan et al., 2013 [4] use a new probabilistic wind power forecasting approach based on the extreme learning machine (ELM), which is a novel learning algorithm proposed for training single-hidden layer feedforward neural networks (SLFNs). It randomly chooses the input weights of hidden layer neurons and analytically determines the output weights through simple matrix computations, therefore featuring an extremely faster learning speed than for most popular learning algorithms such as Back-propagation. ELM has also demonstrated excellent generalization capability and outperformed traditional NNs. In practice, ELM has been used in many different applications, including both regression and classification tasks. Due to the excellent approximation and generalization capabilities, neural networks (NNs) are widely used for wind power forecasts, irrespective of some drawbacks like local minima, overtraining, and high computational costs. Generally speaking, NN-based forecasting methods cannot provide satisfactory predictions if the training data are chaotic or too noisy. Usually the prediction performance cannot be improved by changing the NN structure or increasing the training iteration. To effectively account for forecasting uncertainties, several approaches have been developed to obtain prediction interval (PIs) for NN based methods, including delta, Bayesian, bootstrap, and mean-variance estimation methods. Comparing with other approaches, the bootstrap approach is able to flexibly approximate the non-constant variance and heterogeneous noises thus providing reliable performance, which is used for wind power forecasting recently. In addition, it avoids the calculations of complicated derivatives and the Hessian matrix involved in delta and Bayesian methods. However, due to the limitations of traditional NNs, the bootstrap approach suffers from significantly high computational burden, especially for large datasets. Furthermore, the bootstrap technique for traditional NNs cannot be applicable to the case of ELM, since the associated learning process is very different from that for conventional NN learning algorithms. Therefore, a bootstrap-based ELM approach (BELM) was developed in this paper to construct PIs taking the heteroscedasticity of wind power time series into account. The proposed the BELM method can rapidly formulate the PIs through extremely fast learning by ELM. Notably, though with high extendibility, the work in this paper focuses on a simplified approach with fast speed, using the historical wind power data alone while providing satisfactory performance for hourly ahead and intra-hour forecasting, which is significant for power system operation and control in practice. For instance, in the Nord Pool market in Scandinavia, the hourly market plays a key role in maintaining system balance.

Singh et al.,2007 [5] studied the application of Extreme Learning Machine algorithm for chaotic time series generated by the Mackey Glass delay differential equation with different time delays. They

performed our experiments using *sigmoid*, *sin* and *hardlim* activation functions and demonstrated that the ELM algorithm using *sin* and *sigmoid* activation functions can achieve high prediction accuracy. They concluded that ELM is a promising method for time series prediction problems.

2.2 Ensemble RDLR

Huang et al., 2009 [6] Have shown that efficiently combining multiple weak predictors will create a single strong predictor, this lays the foundation for the concept of ensemble learning. Kumari et al., 2021 [7] suggest that inorder to ensure diversity among the base models, we should emphasize on minimizing correlations between base models, as highly correlated models produce identical output and offer no added value. Accordingly, each base model will have distinct parameters and will be trained on a randomly sampled 80% of the dataset. A highly robust model that

The data preprocessing methods used include Standardization and Principal Component Analysis (PCA). PCA is a well-established technique in data analysis that was first introduced by Pearson in 1901 [8] and later independently developed by Hotelling in 1933 [9]. The fundamental objective of PCA is to reduce the dimensionality of a dataset with multiple interrelated variables, while retaining the maximum amount of variation present in the data.

2.3 TFTs

A hybrid model based on LSTM and CNN was proposed in [10]. The results showed that CNN helped to improve the model performance. [11] developed a method using LSTM and attention mechanism to predict solar power. They found that the proposed method performed better than other available models for the time horizon of short term period.

Wavelet transform (WT) performs better than Emperical Model Decomposition (EMD) for preprocessing the data and GA/PSO were found to be better optimizers than others in [12].

ConvLSTM was introduced in [13] to build an end-to-end trainable model for the precipitation nowcasting problem. The ConvLSTM network [13] captured spatio-temporal correlations better than FC-LSTM and the SOTA (State of the Art) operational ROVER method for precipitation nowcasting problem. [14] proposed a multiple-head convolutional LSTM (MCL) model for healthcare time series classification. MCL is a ConvLSTM with many heads. Thie paper [14] solves the issue of category imbalance in data by normalizing the data, trimming it subsequently and adjusting the weight of sample label along with L2 regularization. MCL could make better judgements since it could extract more features. The paper [15] applied Residual Networks (ResNets), Batch Normalization (BN), Network-in-Network (NiN) along with convLSTM for speech recognition, giving it better generalizations and more expressive power.

[16] develops and ensemble of CNNs (to identify visual patterns) trained over Gramian angular fields (GAF) images, generated from time series related to the Standard and Poor's 500 index future for financial forecasting, where they reported that their model outperformed other trading methods. GAFs are images representing a timeseries in polar coordinate system. Each GAF represents a temporal correlation between each time point. In [17], they introduce the idea of encoding time series into images and dive deeper into GAFs.

In a variety of time series forecasting problems, transformer models have demonstrated cutting-edge performance [18][19][20]. Most notable models, which focus on the less explored and challenging long-term time series forecasting (LTSF) problem, include LogTrans, Informer [20] (AAAI 2021 Best paper), Autoformer, Pyraformer, Triformer and the recent FEDformer [21]. Most significant advantage of transformers over RNN family is parallelism: they can take advantage of GPUs for parallelism (eg: BERT) because of non-sequential processing of input data. Transformers, in particular, are likely the most successful technique for extracting semantic correlations among items in a long sequence. In time series modelling, however, we must extract the temporal relationships from an ordered set of continuous points. The self attention mechanism by itself leads to temporal loss. Experiment findings on nine real-world datasets [21] reveal that LTSF-Linear outperforms existing advanced Transformer-based LTSF models in all circumstances, and frequently by a significant margin. LTSF-Linear was a simple linear model that was used as a baseline for comparison in [21].

Google has recently created a hybrid Transformer-LSTM model that achieves State-of-the-art results in Time series forecasting tasks. The model employs Attention, but it also incorporates an LSTM encoder-decoder stack, which is useful in capturing local temporal relationships. Current research is being focused on improving shortcomings of Transformers, i.e., by incorporating features from different models such as CNNs (eg: vision transformers), RNNs, RL to improve transformers. A Google Research team and the Swiss AI Lab IDSIA suggested a novel architecture called Block-Recurrent Transformer in March 2022. Other competitive models include Google's Temporal Fusion Transformer and Amazon's DeepAR, where both models utilize LSTMs (TFT (Temporal Fusion Transformer) being slightly better).

It is also essential to evaluate a model from practical perspective. Kaggle, which gives indirect empirical evidence on the state of the area of data science, is a solid base for judgement. In the recent Ventilator Pressure Prediction Kaggle competition, multiple-time series problem. The top three teams, as well as many others, used an LSTM-based component in their final solution (e.g. stacked LSTMS, Bi-directional LSTMS). The winning team proposed a multi-level deep architecture that featured an LSTM network and a Transformer block, among other things.

SCINet is another model proposed by [22] that gives SOTA results on some datasets for time series forecasting. This conducts sample convolution and interaction for temporal modeling and applies it for the time series forecasting problem. The proposed 'downsample-convolve-interact' architecture allows for multi-resolution analysis while also broadening the receptive area of the convolution operation, making it easier to extract temporal connection information with improved predictability.

3 Methodology

3.1 Data Collection using Weather Research and Forecasting (WRF)

WRF [23] is a meso-scale numerical weather prediction model, which can be used to produce realistic meteorological data over an entire province, even where no nearby observations exist. Many national meteorological centres, laboratories, universities, and companies around the world use this model on regular basis to obtain weather data. It gives hourly, three-dimensional, grid-structured, meteorological data, called WRF data. The following steps are followed to obtain the WRF data:

1. Define the area (spatial) and time (temporal) period of data to be downloaded and processed.
2. Download the complete WRF data-generating package for Linux-based computers, after the spatial and temporal boundaries of the WRF data are specified.
3. Initialise the WRF pre-processing system (WPS), which is responsible for preparing input to WRF for real-data simulations.
4. Define the simulation coarse domain and nested domains and compute latitude, longitude, map scale factors, and Coriolis parameters at every grid point within the defined area.
5. Run the WRF for real or ideal cases depending on the requirement.
6. Obtain the WRF output (over 215 variables or features) and select the most relevant and high correlation (both positive and negative) features as inputs to the models.

Figure 1 depicts the structure of WRF along with its different sub-components. The Geogrid program present in WPS, defines the Map projection, Geographic location and the Dimensions of domains.

3.1.1 Function of Each WPS Program

The WPS consists of three independent programs: geogrid, ungrib, and metgrid. Also included in the WPS are several utility programs, which are described in the section on utility programs. A brief description of each of the three main programs is given below, with further details presented in subsequent sections.

Program geogrid

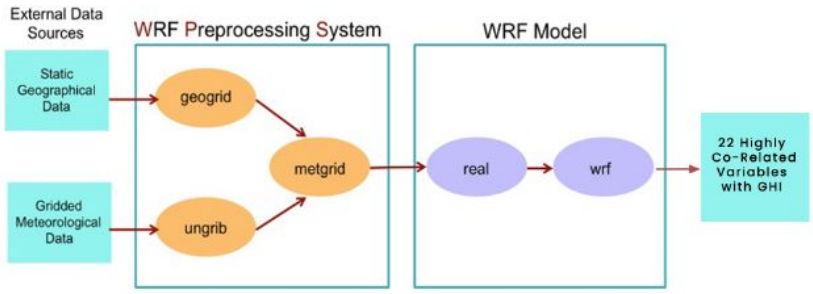


Figure 1: Weather Research and Forecasting Base Structure

The purpose of geogrid is to define the simulation domains, and interpolate various terrestrial data sets to the model grids. The simulation domain is defined using information specified by the user in the "geogrid" namelist record of the WPS namelist file, namelist.wps. By default, and in addition to computing latitude and longitudes for every grid point, geogrid will interpolate soil categories, land use category, terrain height, annual mean deep soil temperature, monthly vegetation fraction, monthly albedo, maximum snow albedo, and slope category to the model grids. Besides interpolating the default terrestrial fields, the geogrid program is general enough to be able to interpolate most continuous and categorical fields to the simulation domains. New and additional data sets may be interpolated to the simulation domain through the use of the table file, GEOGRID.TBL. The GEOGRID.TBL file defines each of the fields that will be produced by geogrid; it describes the interpolation methods to be used for a field, as well as the location on the filesystem where the data set for that field is located. Output from geogrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, geogrid can be made to write its output in NetCDF for easy visualization using external software packages, including ncview and RIP4.

Program ungrib

The ungrib program reads GRIB files, degribs the data, and writes the data in a simple format, called the intermediate format. The GRIB files contain time-varying meteorological fields and are typically from another regional or global model, such as NCEP's NAM or GFS models. The ungrib program can read GRIB Edition 1 and GRIB Edition 2 files. GRIB files typically contain more fields than are needed to initialize WRF. Both versions of the GRIB format use various codes to identify the variables and levels in the GRIB file. Ungrib uses tables of these codes – called Vtables, for variable tables – to define which fields to extract from the GRIB file and write to the intermediate format. Vtables for common GRIB model output files are provided with the ungrib software. Vtables are available for NAM 104 and 212 grids, the NAM AWIP format, GFS, the NCEP/NCAR Reanalysis archived at NCAR, RUC (pressure level data and hybrid coordinate data), and AFWA's AGRMET land surface model output. Users can create their own Vtable for other model output using any of the Vtables as a template; further details on the meaning of fields in a Vtable are provided in the section on creating and editing Vtables. Ungrib can write intermediate data files in any one of three user-selectable formats: WPS – a new format containing additional information useful for the downstream programs; SI – the previous intermediate format of the WRF system; and MM5 format, which is included here so that ungrib can be used to provide GRIB2 input to the MM5 modeling system.

Program metgrid

The metgrid program horizontally interpolates the intermediate-format meteorological data that are extracted by the ungrib program onto the simulation domains defined by the geogrid program. The interpolated metgrid output can then be ingested by the real.exe program. The range of dates that will be interpolated by metgrid are defined in the "share" namelist record of the WPS namelist file, and date ranges must be specified individually in the namelist for each simulation domain. Since the work of the metgrid program, like that of the ungrib program, is time-dependent, metgrid is run every time a new simulation is initialized. Control over how each meteorological field is interpolated is provided by the METGRID.TBL file. The METGRID.TBL file provides one section for each field, and within a section, it is possible to specify options such as the interpolation methods to be used for the field, the

field that acts as the mask to be used for masked interpolations, and the staggering (e.g., U, V in ARW; H, V in NMM) to which a field is to be interpolated. Output from metgrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, metgrid can be made to write its output in NetCDF for easy visualization using external software packages, including RIP4.

3.1.2 Function of Each WRF Program

The WRF software package consists of several executables, including real and wrf. These executables are responsible for different stages of the modeling process.

Program Real

real.exe is short for "real data initialization". This executable is responsible for initializing the model with real-world atmospheric data. It reads in observed data from a variety of sources, such as weather stations, radars, and satellites, and interpolates the data onto the model's grid. This process is important because it allows the model to start from a state that is close to the real-world conditions.

Program WRF

wrf.exe is short for "WRF model execution". This executable is responsible for running the actual simulation. It takes the initial conditions generated by real.exe and uses them to simulate the behavior of the atmosphere over a specified time period. The model solves a set of values that describe the dynamics of the atmosphere, as well as the exchange of heat, moisture, and other physical properties.

Together, real.exe and wrf.exe form the core of the WRF modeling system. They allow users to simulate and forecast weather with a high degree of accuracy and precision, making them valuable tools for a wide range of applications in the fields of meteorology, atmospheric science, and climate research.

3.2 Data Preprocessing

3.2.1 Extreme Learning Machine Data Preprocessing

The Extreme Learning Machine (ELM) Model used a single time series variable as its input. The data obtained from WRF contained over 215 variables from which the Global Horizontal Irradiance (GHI) variable was selected as the input to the ELM Model. The following steps were carried out to preprocess the data to make it compatible with the ELM Model:

1. **Data Normalisation**: The raw data obtained varied between large positive and negative values. Such values may cause an unintended and erroneous bias towards them during the training process. Hence, all of the data was normalised to the range $[-1, 1]$.

2. **Sliding Window Conversion**: The Time Series data by itself represents the data in a time serial fashion which is not in supervised form as required by the ELM Model. Hence, the series data was converted into a Supervised Time Series by using the Sliding Window Technique.

3.2.2 Ensemble RDLR

The data obtained from WRF contained samples collected in intervals of 5 minutes. The data is downsampled by 0, 1, 3, 6 and 18 points or 0, 5, 15, 30 and 90 minutes respecively to meet the short-term forecasting periods, before passing it through the data pre-processing pipeline. The pre-processing pipeline contains two steps:

1. **Standardisation**: The raw data varied over a wide range of values which may cause the system to get biased to higher magnitude values. Hence, the data was standardised to remove these anomalies due to the sheer magnitude of values and focus more on underlying patterns. Mathematically standardisation is carried out as,

$$z_i = (x_i - \bar{x})/\sigma \tag{1}$$

, where $\bar{x}$ is the mean value of the data, x_i is the i^{th} component of the scaling data and σ is the unit variance.

2. **Principle Component Analysis (PCA)**: PCA is a dimensionality reduction technique that engineers new variables that are a linear combination of existing features and orders them based on how well they explain the variation of data. The new variables are solely for explaining the variability and have no physical interpretation. The input data from WRF contained all features having different degrees of correlation with the target variable and with other input features. Due to this nature of the data, PCA was carried out with all the features and the dimensionality of the data was reduced from over 215 features to a few dimensions.

3.2.3 Temporal Fusion Transformer Preprocessing

The input and the target dataframe have been concatenated into one dataframe, `df`. The column names have been renamed to "FT-1", "FT-2", ..., "FT-23". The data needs to be converted to `TimeSeriesDataset` format where we specify which features are static and time varying. CUrrently, the features are considered to be time varying rather than static since we dont have enough information about them.

Consider the current timestep as t, lookback window m, step ahead window n. The following will be required by the model: Observed past input x [t-m, .. t], Future known inputs x [t+1 .. t+n], set of static variables s (if present), target variable y [t+1 .. t+n].

3.3 Extreme Learning Machines

The learning speed of the neural networks (feed-forward, in particular) is relatively slower than the required speed for fast pace performance. This has bottle-necked several applications over the past decade and the conventional slow-paced gradient-based training algorithms and iterative tuning of hyper-parameters can be identified as the key reasons for it. Extreme Learning Machine is an algorithm for Single Layer Feed-forward Neural Networks (SLFNs) which aims to tackle these problems [3].

The ELM algorithm can adaptively set the number of hidden layer nodes. The ELM also randomly assigns the input weights and hidden layer biases. The output layer weights are obtained by the least square method making the whole learning process iteration-free. Hence the training time of ELMs is vastly smaller than traditional back-propagated neural networks [24].

Similar to [5], we implemented the ELM algorithm for SLFNs and applied it to the time series forecasting problem. The input to the ELM Model was the GHI variable from WRF. Due to the small training time of the ELM Model, we run a brute force search to find the best ELM Model for each forecasting time step. It was observed that the ELM Model's performance is vastly dependent on the configuration of the Sliding Window applied to the input data.

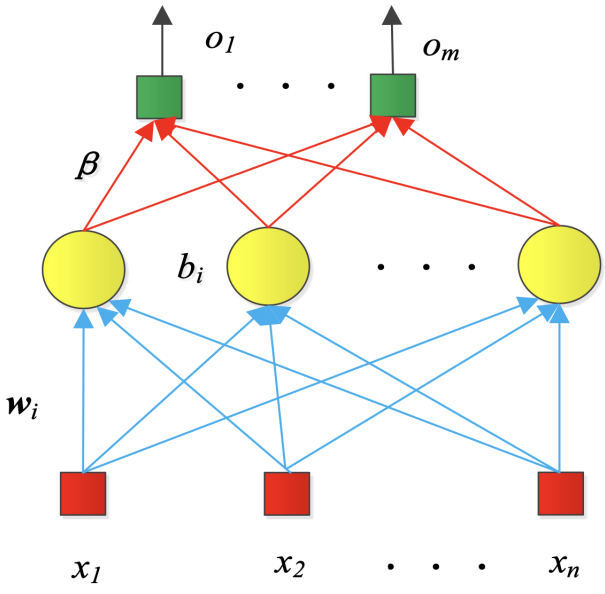


Figure 2: Extreme Learning Machines Structure Graph [1].

According to Wang et al., 2022 [1], a training set is defined as,

$$S = \{(x_i, t_i) | x_i = (x_{i1}, x_{i2}, ..., x_{in})^T \in R^n, t_i = (t_{i1}, t_{i2}, ..., t_{in})^T \in R^m\}$$

where x_i denotes the input value and t_i represents the target value. Using this training set the output o of an ELM with $\hat{N}$ hidden neurons can be expressed as,

$$\sum_{i=1}^{\hat{N}} \beta_i g(w_i x_j + b_i) = o_j \quad \forall \ j \in [1, N] \tag{2}$$

Where $g(x)$ means the activation function in the hidden layer. In ELM, activation functions are nonlinear ones to provide nonlinear mapping for the system.

$$X = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_{\hat{N}^T} \end{bmatrix}_{\hat{N} \times m}, T = \begin{bmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_{\hat{N}^T} \end{bmatrix}_{\hat{N} \times n}$$

Therefore, training the SLFN is to find the best w_i, b_i and β_i.

3.4 Ensemble Ensemble RDLR

The Ensemble Model is a model generated after combining individual sub-models which independently forecast the target and their results are combined using another sub-model to get the final value of the forecasted target variable. This collaborative unit model helps in understanding the data in all aspects as each sub-model is bound to be sensitive to different patterns present in the input data. Correlation between the base models has to be minimized as highly correlated models will just produce the same output for each model, which has no diversity and will be of no use [7]. The Ensemble Model contained the following sub-models:

1. **Random Forest Regression (RF)**: This sub-model works by using a group of decision trees, each behaving as an independent weak predictor. The decision tree provides a vote as to what the predicted value should be and the majority vote is taken as the output for a given forest, while for regression the average of all trees is computed as the final result. This combining multiple outputs boosts the performance of the individual decision trees [25]. Some highlights of the random forest model include random feature selection, bootstrap sampling, out-of-bag error estimation and full-depth decision tree growing [26]. With these attributes, it can model relatively complex interaction between input variables and is quite robust with respect to outliers. Figure 3 shows the structure of the RF sub-model as explained above.

2. **Deep Neural Networks (DNN)**: These are the standard artificial neural networks (ANNs) that contain more than one hidden layer making it well suited for modeling non-linear relationships. The number of hidden layers and the number of neurons per hidden layer are treated as hyper parameters that can be tuned as per required accuracy. However, DNNs act on fixed length vector inputs and cannot process time series data directly. Therefore the time series data needs to be converted to a format that is actually useful for the DNN. This can be done through data compression. Generally there are four approaces for using DNNs with time series data: restricted Boltzmann machine (RBM), deep belief network (DBN), autoencoder (AE) and deep convolutional neural network (DCNN). Among these, we'll use autoencoders for data compression as they can learn generative models of data and filter out noise [27] which will aid in forecasting of stochastic information like weather data.

   To enable this conversion the LSTM Auto-encoder (Auto-LSTM) implementation of the auto-encoder will be the utilized.The Auto-LSTM approach uses an encoder-decoder architecture to compress the time series data in a single vector of fixed length, which is then passed onto a regular neural net for training and prediction. The quality of the Auto-LSTM network is assessed by the decoder's ability to successfully recreate the input time series data from the encoder's compressed data, the principle being to minimize the reconstruction error [28]. Once training is done the decoder is discarded and the only the encoder is retained for converting any input to a fixed length vector. The encoder-decoder architecture of Auto-LSTM can be implemented using 3 types of layers:

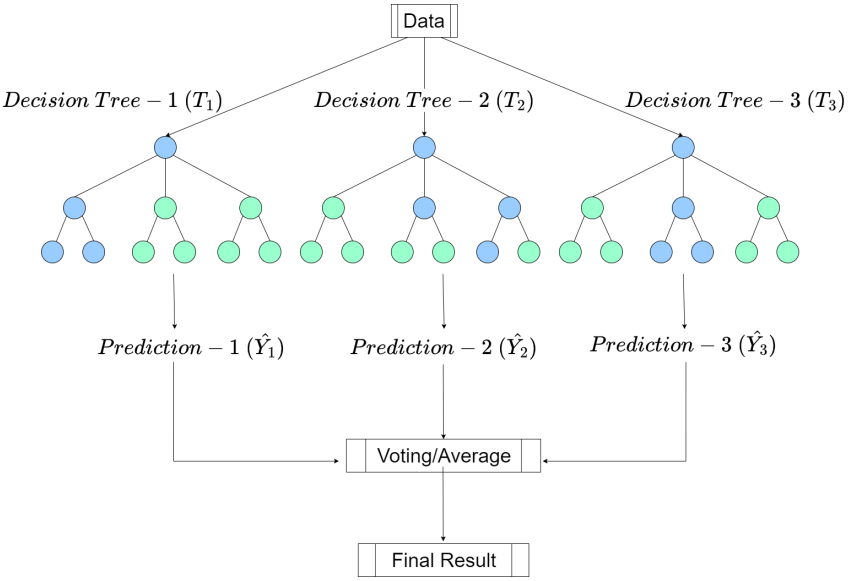


Figure 3: Random Forest Regression Sub-Model Structure

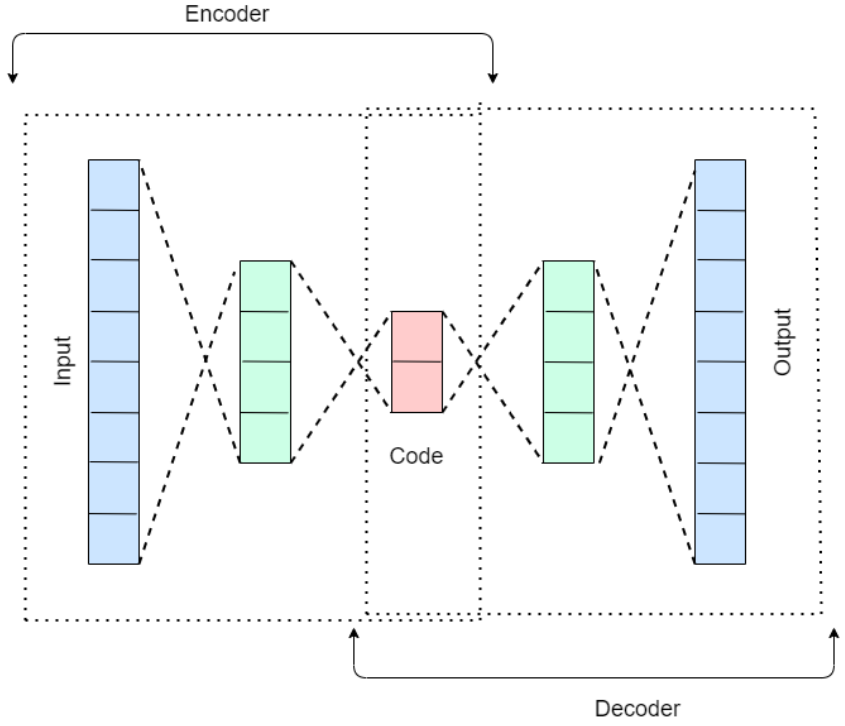


Figure 4: Auto-LSTM Structure

3. **LSTM**: RNNs are neural networks that are unrolled in time where each network passes information from the previous iteration (past) to the current iteration. This allows the network to retain information from the past and use it to make better decisions in problems where a subject of the past might be referred again in the future. However RNNs do not perform well with long term dependencies in practice, leading to the development of it's variants. LSTM's are one such variant of RNN which have the ability to forget unnecessary information and only pass on only the required information to the next round of operations. This is done through the use of maintaining a state vector which stores information from previous steps and a series of operations that decide which information needs to be forgotten or updates for the future. These operations are carried out with the help of gates, which consist of a sigmoid neural net to decide which values to operate on followed by pointwise multiplication to carry out the actual update.

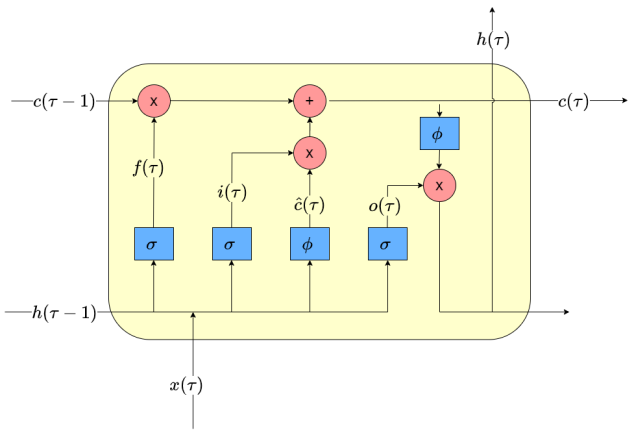


Figure 5: LSTM Architecture

4. **Ridge Regression**: All the models in the ensemble model are combined using the Ridge Regression (RR) Model. It is a modified version of the Ordinary Least Squares (OLS) method, where a bias (or) penalty is added to the loss function. The bias is introduced as the sum of slopes, weighted with a parameter (generally λ or α). The slope of the target variable is considered with respect to a predictor variable. The bias term desensitises the dependence of the target to change in any one of the predictor variables as much as possible. This essentially reduces error and over fitting in datasets having some degree of multi co-linearity among predictor variables [29]. The penalty term also helps in cases where number of data points is lesser than the number of co-efficients to be solved for.

   In this architecture The $n+2$ base models produce $n+2$ predicted outputs as follows:

$$\begin{bmatrix} \widehat{y}_1 = f_1(X) \\ \widehat{y}_2 = f_2(X) \\ \cdot \\ \widehat{y}_{n+2} = f_{n+2}(X) \end{bmatrix} \tag{3}$$

   These outputs have to be combined together get a single result, which will be the weighted sum of all the n weak predictor outputs. This will be done by the final ridge regressor model. As RR is used to ensemble the base models, we will refer to it as Ensemble RDLR (RF-DNN-LSM-RR) here on to differentiate it from the base models.

$$\widehat{y} = w_0 + w_1 * \widehat{y}_1 + w_2 * \widehat{y}_2 + \ldots\ldots + w_{n+2} * \widehat{y}_{n+2} \tag{4}$$

   Here w_i is the weight assigned to each of the base models which can be obtained by training the Ridge regressor using the following cost function:

$$C(w) = \frac{1}{2(n+2)} \sum_{i=1}^{n+2} (y_i - \widehat{y}_i)^2 + \lambda \sum_{j=1}^{n+2} w_j^2 \tag{5}$$

Each base model in the architecture also requires specialized hyper parameter tuning techniques. These are described briefly below.

- **Random Forest**: Random search and grid search are the most commonly used techniques for tuning random forests [30]. Random search involves randomly evaluating a subset of all possible combinations of input parameters (search space), to get an approximate idea of the most effective combination of parameters. It is followed by Grid Search which fine-tunes the parameters obtained through random search. This is done by setting up a new search space closely centred around the approximate values obtained from the random search. This technique was used to tune the Random Forest Sub-model.

- **DNN and LSTM**: Both Auto-LSTM and LSTM models use LSTM layers in their respective implementations. Hence, we'll use Particle Swarm Optimisation (PSO) to tune the LSTM layers in both models.
- **Ridge Regression**: Similar to [31], we used Bayesian optimisation will be used for the tuning of the Ensemble RDLR model.

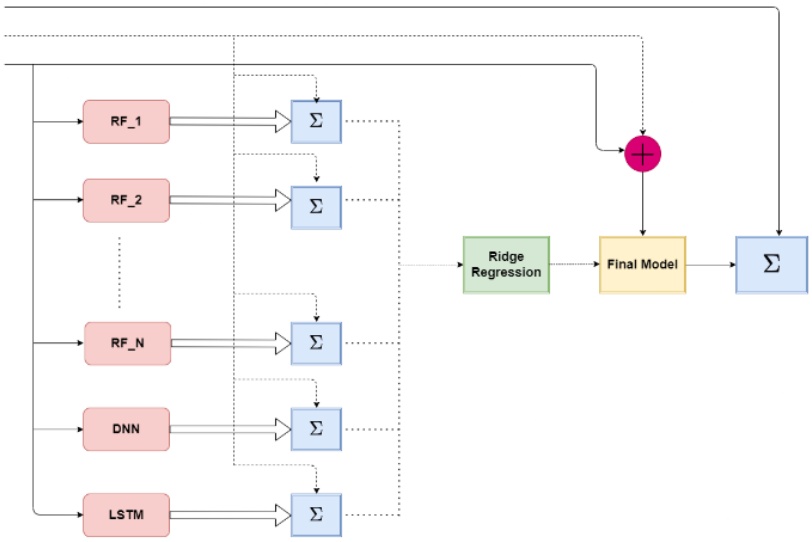


Figure 6: Overall Ensemble RDLR Model Architecture

3.5 Temporal Fusion Transformer

Temporal Fusion Transformer is a diverse and high-performing model. The main advantage of it is providing interpretability. Explainability is increasingly a primary issue, particularly in production. In certain circumstances, explainability takes precedence above accuracy.

3.5.1 Architecture of TFT

Temporal Fusion Transformer's architecture incorporates various major developments from the Deep Learning area while also proposing some novelties of its own. The processing is done in two parts: local and global processing. Local processing focuses on characteristics of specific events, while global processing focuses on aggregating the characteristics of all time series. The model proposes a unique Muti Head attention mechanism that, when evaluated, provides additional knowledge on feature importances by utilising self-attention.

The architecture of temporal fusion transformer is given in the figure [7]

3.5.2 Model Components

- **Gated Residual Network (GRN):** assist the network in determining which input modifications are straightforward and which call for more intricate modelling. GRN can also make use of static variables and skip input entirely.
- **Variable Selection Network (VSN):** It is like a feature selection mechanism. For three types of inputs (shown in different colors in the architectural model), TFT (Temporal Fusion Transformer) uses 3 instances of the VSN. VSN uses GRN for its filtering capabilities.
- **LSTM Encoder Decoder Layer:** After the input has passed through VSN, it gets properly encoded and the features get weighted. This layer helps in using the sequential ordering, by producing context-aware embeddings, similar to positional encoding used in the classic Transformer. The paper used LSTM encoder decoder in order to consider all types of input. The known inputs are fed into the encoder, while the unknown future inputs are fed into the decoder.

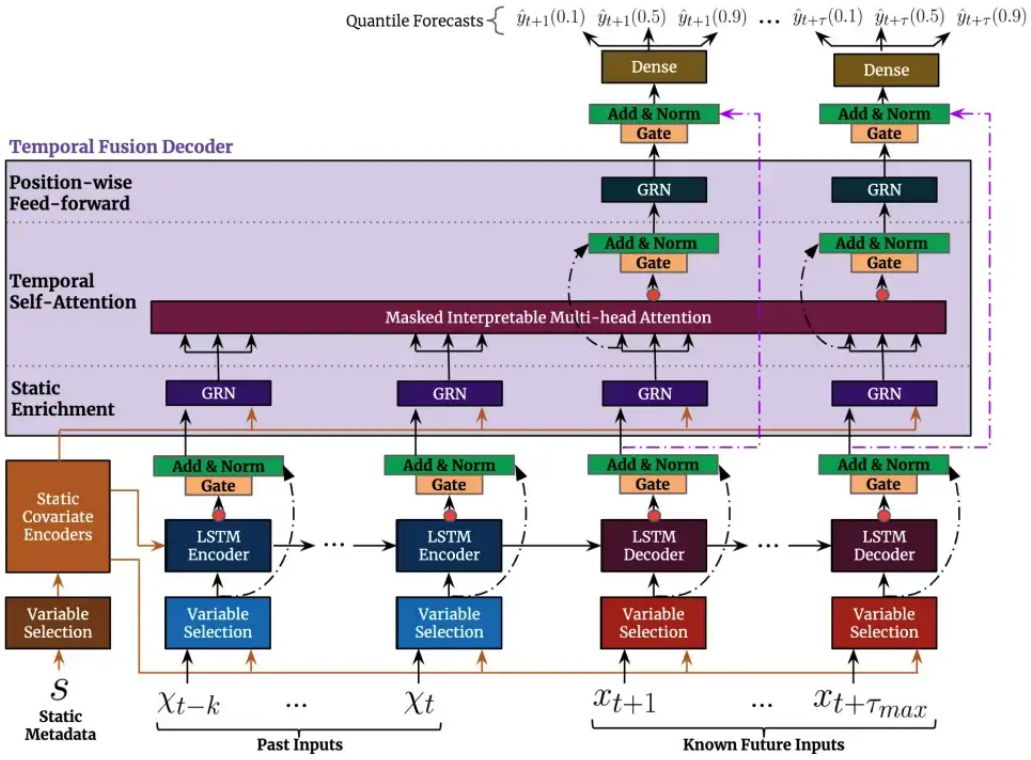


Figure 7: Architecture of the Model

- **Interpretable Multi-Head Attention:** provides feature interpretability in contrast to conventional implementation in traditional transformers (V/K/Q matrices).
- **Quantile Regression:** When the assumptions of linear regression are broken, quantile regression, an extension of standard linear regression, can be used to estimate the conditional median of the target variable. Quantile regression may compute the 0.25 and 0.75 quantiles in addition to the median (or any percentile, for that matter), thus the model can produce a prediction range around the actual prediction.

3.6 Model Performance Evaluation Metrics

All of the models used the following performance evaluation metrics to standardise the process and compare results.

- Mean Square Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{6}$$

- Root Mean Sqaure Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} = \sqrt{MSE} \tag{7}$$

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y_i} - y_i| \tag{8}$$

- Coefficient of Determination (R^2):

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y_i} - \bar{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{9}$$

Here y_i is the actual value, $\hat{y}_i$ is the predicted value from the model and $\bar{y}_i$ is the mean value of all the actual values.

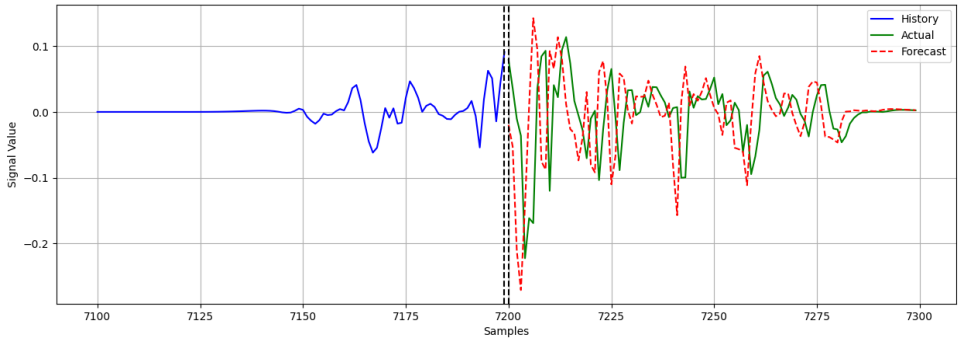
4 Results

4.1 Extreme Learning Machines

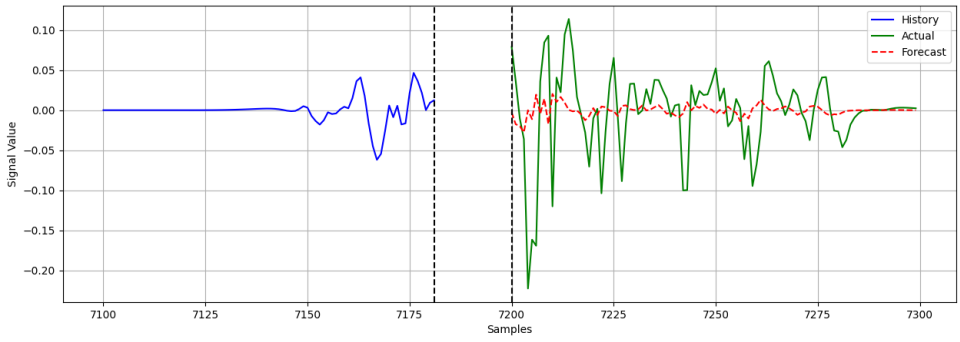
The ELM Model was tested using a traditional 80-20 training-testing split of the input data. Table 1 shows the ELM Model's performance for each of the forecasting time horizons. Figure 8 shows the model's forecast plot for T+0 (i.e. immediate) and T+18 (90 minutes ahead) forecasts.

Table 1: Performance Comparison of ELM Model across multiple forecasting time horizons

Time (min)	**RMSE**	**MAE**	**R^2 (%)**
0	0.0203	0.0063	78.0397
5	0.0284	0.0088	57.2428
15	0.0427	0.0157	3.8489
30	0.0441	0.0162	-2.4288
90	0.0411	0.0137	11.7849



(a) Extreme Learning Machine's Immediate (T+0) Forecast



(b) Extreme Learning Machine's 90 minutes ahead (T+18) Forecast

Figure 8: Extreme Learning Machine's Forecasts

4.2 Ensemble RDLR

The training was carried out on a home laptop computer with an Intel® Core(TM) i5-10300H CPU clocked at 2.50GHz and 8.00 GB RAM. Given the nature of the Ensemble RDLR model, the tests and comparisons were carried out on two fronts:

1. **Comparison across the Base Models**: The objective of this type of test is to check the relative performance of each of the base models (Random Forest, Auto-LSTM DNN and LSTM) with the overall ensemble (Ensemble RDLR) for a given forecasting time horizon. Any time horizon can be selected as the focus is mainly to find the relative accuracy and forecasting errors of the base models. Table 2 highlights this comparison for immediate forecasting time horizon (i.e. t+0 forecast) to better identify the differences between each of the models.

2. **Comparison across different forecasting time horizons**: The objective of this type of test is to check the performance of the Ensemble RDLR model (i.e. collective of all the base models) for different short-term forecasting time horizons. For this test, we used forecasting time horizons ranging from 0 minutes (immediate) to 90 minutes ahead, to understand the Ensemble RDLR model's performance in greater detail. Table 4 tabulates the results obtained for the different forecasting time horizons.

Table 2: Performance Comparison across Base Models at T+0 forecasting time horizon

Model	**RMSE**	**MAE**	**R^2 (%)**
RF	0.00113	0.00047	99.944
Auto-LSTM	0.00096	0.00034	99.959
LSTM	0.00099	0.00038	99.957
Ensemble RDLR	0.00094	0.00032	99.960

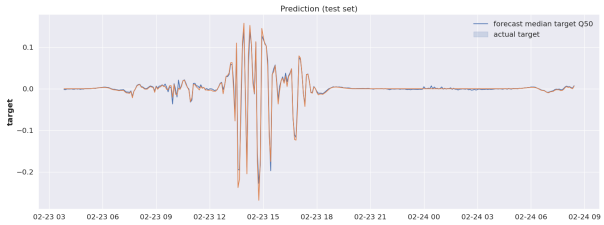
Table 3: Performance Comparison of Ensemble RDLR Model across multiple forecasting time horizons

Time (min)	**RMSE**	**MAE**	**R^2 (%)**
0	0.00094	0.00032	99.960
5	0.03952	0.01561	30.901
15	0.04795	0.02053	1.943
30	0.04706	0.01975	4.878
90	0.04898	0.02082	0.222

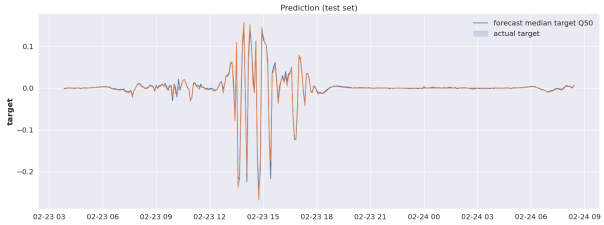
The above Tables 2 and 4 show the nature of Ensemble RDLR model for the input data. It was observed that each of the base models performed similarly at any given time horizon, according to the first comparison test. The Ensemble RDLR model, collectively, quickly loses performance as the forecasting time horizon increases as observed from the R^2 values in Table 4. Another important aspect noted from these tests reflected that the error values themselves are relatively small compared to the performance drops faced by the Ensemble RDLR model, suggesting that the forecast obtained by the Ensemble RDLR model is closer to the mean of the original data but fails to predict the exact nature of the data. Forecast Horizons 5 minutes to 90 minutes explain this fact because the increase in error values (RMSE and MAE) is very low even though the drop in R^2 is very high.

4.3 Temporal Fusion Transformer

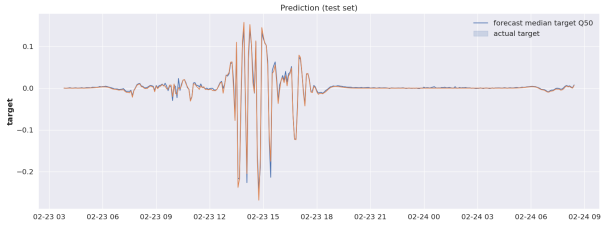
The percentages for which we want to obtain predicted forecast bands have been obtained for the following pairs: 10%-90%, 20%-80%, 1%-99%, and the central value (median). The plots in Figure 9 have been obtained from the 50% quantile or the median. This central value is nothing but the point estimate which could have been obtained from a deterministic forecast model.



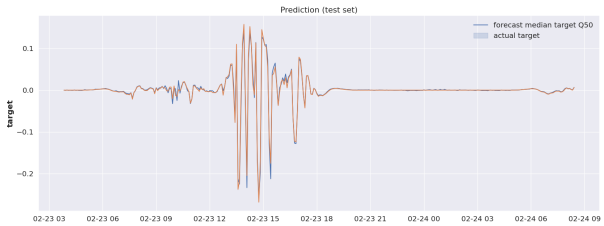
(a) TFT's Immediate (T+0) Forecast



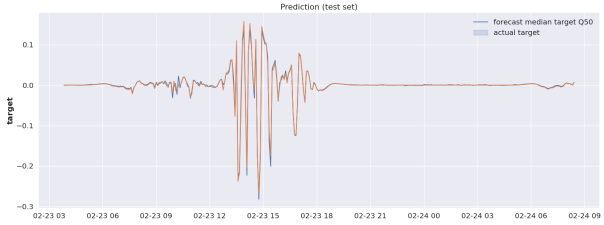
(b) TFT's 5 minutes ahead (T+1) Forecast



(c) TFT's 15 minutes ahead (T+3) Forecast



(d) TFT's 30 minutes ahead (T+6) Forecast



(e) TFT's 90 minutes ahead (T+18) Forecast

Figure 9: TFT's Forecasts

The error values in Table 4 have been obtained for 50% quantile.

Table 4: Performance Comparison of TFT Model across multiple forecasting time horizons

Time (min)	**RMSE**	**MAE**	**R^2 (%)**
0	0.0060590	0.0028667	97.83451
5	0.0053408	0.0025305	98.31745
15	0.0053867	0.0027578	98.28837
30	0.0054405	0.0023579	98.25402
90	0.0048140	0.0022145	98.63300

5 Conclusion

In this paper, we observed the following outcomes. The ELM Model obtained a maximum R^2 score of 78% for the immediate forecasting time horizon but dropped to 11.78% for 90 minutes ahead, being the best model of the models compared in this paper. Comparison case 2 from Ensemble RDLR Model reflects that, overall model performance decays as we increase the time horizon for prediction. The highest accuracy is for 0-minute ($R^2 = 99.96\%$) and 5-min ($R^2 = 30.90\%$) minute forecast followed by others in 15, 30 and 90 minute horizons. The rapid decrease in accuracy can be attributed to the stochastic nature of weather and potential inconsistencies in the data. We can observe that TFT (Temporal Fusion Transformer) offers good performance and most importantly, provides feature interpretability and explainability. The TFT model was obtained from a standard optimized implementation, due to which he results are good. According to Gartner, this is one of the directions that deep learning will take in the future. Using quantile regression instead of Linear regression helps us when assumptions of linear regression are not met.

Acknowledgements

It has been a great pleasure to work with different individuals whose perspective and ideas has directly or indirectly assisted or motivated us. We would like to thank everyone whose ideas lead to this completion of the project. It is a matter of fact a great privilege for us to acknowledge their assistance and contributions to our project.

First and foremost, we would like to express our sincere gratitude towards Dr. Yashwant Kashyap, our major project supervisor under whose supervision there was successful completion of our project. Without his invaluable guidance and suggestions, it would have been a difficult journey for us. His useful suggestions for this whole work and cooperative behavior are sincerely acknowledged.

We would like to thank the Department of Electrical and Electronics Engineering for adding this major project as part of the final year curriculum and hence giving us this opportunity to undertake the project. The great need for research, time and sheer coding has allowed us to harness our skills, experience and knowledge.

We would like to thank and express our gratitude to all our respective subject teachers for sharing their precious knowledge, constant support and guidance.

Last but not the least, we would like to thank our friends for motivating us and providing numerous assistance throughout the project development.

Future Work

Future improvements to the Ensemble RDLR model may involve enhancing its ability to make long-term predictions. One approach that holds promise is the use of Heap-based and Particle Swarm Optimization techniques. These methods have shown effectiveness in the context of weather data forecasting and could potentially be adapted to further enhance the Ensemble RDLR model's forecasting capabilities for long-term predictions. Another possible improvement includes automating the WRF software to extract real-time data for any given coordinates. There's scope for hardware acceleration through FPGA and edge computing with the help of micro-controllers. LSTM implementations using both PyTorch, Tensorflow for different time windows and LSTM model from scratch have been implemented. The future work lies on exploring and applying pervasive attention (as it can be even better than transformers for seq to seq tasks) on TFT, studying SCINet, applying this and benchmark dataset for TFT and other SOTA (State of the Art) methods

A Appendix

A.1 List of Abbreviations

ELM	Extreme Learning Machines
RF	Random Forest
LSTM	Long Short-Term Memory
Auto-LSTM	Autoencoder Long Short-Term Memory
RR	Ridge Regression
Ensemble RDLR	RF-DNN-LSTM-RR
NN	Neural Network
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
R^2	Co-efficient of determination
PI	Prediction Intervals
TFT	Temporal Fusion Transformer

References

[1] J. Wang, S. Lu, S.-H. Wang, and Y.-D. Zhang, "A review on extreme learning machine," *Multimedia Tools and Applications*, vol. 81, no. 29, pp. 41611–41660, 2022.

[2] D. Shalaeva, O. Kukartseva, V. Tynchenko, V. Kukartsev, S. Aponasenko, and E. Stepanova, "Analysis of the development of global energy production and consumption by fuel type in various regions of the world," in *IOP Conference Series: Materials Science and Engineering*, vol. 952, p. 012025, IOP Publishing, 2020.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006. Neural Networks.

[4] C. Wan, Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong, "Probabilistic forecasting of wind power generation using extreme learning machine," *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1033–1044, 2013.

[5] R. Singh and S. Balasundaram, "Application of extreme learning machine method for time series analysis," *International Journal of Computer and Information Engineering*, vol. 1, no. 11, pp. 3407–3413, 2007.

[6] F. Huang, G. Xie, and R. Xiao, "Research on ensemble learning," in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 3, pp. 249–252, IEEE, 2009.

[7] P. Kumari and D. Toshniwal, "Extreme gradient boosting and deep neural network based ensemble learning approach to forecast hourly solar irradiance," *Journal of Cleaner Production*, vol. 279, p. 123285, 2021.

[8] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[9] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[10] K. Wang, X. Qi, and H. Liu, "A comparison of day-ahead photovoltaic power forecasting models based on deep learning neural network," *Applied Energy*, vol. 251, p. 113315, 2019.

[11] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, and Y. Du, "Short-term photovoltaic power forecasting based on long short term memory neural network and attention mechanism," *Ieee Access*, vol. 7, pp. 78063–78074, 2019.

[12] P. Singla, M. Duhan, and S. Saroha, "A comprehensive review and analysis of solar forecasting techniques," *Frontiers in Energy*, vol. 16, no. 2, pp. 187–223, 2022.

[13] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[14] Y. Wang, L. Sun, and D. Peng, "A multihead convlstm for time series classification in ehealth industry 4.0," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[15] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4845–4849, IEEE, 2017.

[16] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, "Deep learning and time series-to-image encoding for financial forecasting," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 683–692, 2020.

[17] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, 2015.

[18] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *arXiv preprint arXiv:2001.08317*, 2020.

[19] J. Grigsby, Z. Wang, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," *arXiv preprint arXiv:2109.12218*, 2021.

[20] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11106–11115, 2021.

[21] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," *arXiv preprint arXiv:2205.13504*, 2022.

[22] M. Liu, A. Zeng, Z. Xu, Q. Lai, and Q. Xu, "Time series is a special sequence: Forecasting with sample convolution and interaction," *arXiv preprint arXiv:2106.09305*, 2021.

[23] W. Wang, D. Barker, J. Bray, C. Bruyère, D. Michael, J. Dudhia, D. Gill, and J. Michalakes, *User's Guide for Advanced Research WRF (ARW) Modeling System Version 2.2*. Mesoscale & Microscale Meteorology Division, National Center for Atmospheric Research, January 2007.

[24] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artificial Intelligence Review*, vol. 44, pp. 103–115, 2015.

[25] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and qsar modeling," *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[26] R. Jiang, W. Tang, X. Wu, and W. Fu, "A random forest approach to the detection of epistatic interactions in case-control studies," *BMC bioinformatics*, vol. 10, no. 1, pp. 1–12, 2009.

[27] R. Ahmed, V. Sreeram, Y. Mishra, and M. Arif, "A review and evaluation of the state-of-the-art in pv solar power forecasting: Techniques and optimization," *Renewable and Sustainable Energy Reviews*, vol. 124, p. 109792, 2020.

[28] B. Hou, J. Yang, P. Wang, and R. Yan, "Lstm-based auto-encoder model for ecg arrhythmias classification," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1232–1240, 2019.

[29] H. Duzan and N. S. B. M. Shariff, "Ridge regression for solving the multicollinearity problem: review of methods and models," *Journal of Applied Science*, 2015.

[30] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, vol. 9, no. 3, p. e1301, 2019.

[31] A. Stuke, P. Rinke, and M. Todorović, "Efficient hyperparameter tuning for kernel ridge regression with bayesian optimization," *Machine Learning: Science and Technology*, vol. 2, no. 3, p. 035022, 2021.