People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering
Department of Electronics

Project Report Presented in Partial Fulfilment of
the Requirements of the Degree of

'Master'

In Computer Engineering

Title:

Time Series Forecasting using Transformer for Solar Power Prediction

Presented By:

- **Manel Hamadou**
- **Line Sarra Benchemam**

Supervisor:

Dr. Elhocine Boutellaa

Registration Number:…/2024

Dedication

I would like to express my heartfelt dedication to my parents, who have supported me unconditionally through every step of my academic journey. To my aunt Esma, who has been a constant source of maternal guidance and support in my life. I am deeply grateful to my aunts and uncles for their love, guidance, patience, and belief in my abilities. They have stood by me, never doubting my potential. I also extend my sincere appreciation to my siblings Hanny and Nouha for their support.

My heartfelt thanks go to my friends, whose support formed the foundation of my success. Special mention to Manel, Zola Hanane, Amani, and others who have supported me in countless ways. Though I cannot name everyone individually, please know that your presence in my life means everything to me. Your belief in me, even when I doubted myself, has been priceless. I am fortunate to have crossed paths with each of you.

I also acknowledge and thank myself for persevering through tough times, even when the purpose of my efforts was in question and I felt lost. Your resilience has brought me to this point, and I can say that I'm proud of the person I've become today.

I honor my late grandfather and uncles, particularly Uncle Aziz. Although they are no longer with us, I know they would have taken immense pride in my achievements, and their absence will be deeply felt.

To my wonderful partner, who has been an absolute blessing throughout this project. Your unwavering support and invaluable assistance were indispensable in achieving its successful completion, significantly contributing to my academic advancement and personal growth. I am profoundly grateful to you for everything.

Line Sarra Benchemam

Dedication

To my dearest parents, Terfi Ouerdia and Hamadou Aomar,
Your unconditional love and endless sacrifices have paved the way for my success. Your trust in my abilities, even during the darkest times, has been my guiding star. This work is as much yours as it is mine, for without your constant support, I would not have come this far.

To my amazing sister and brothers, Sihem, Wassim, Fouad, Rabah,
Your encouragement and understanding have been a source of immense strength. Thank you for always being there, for listening to my frustrations, and for celebrating my victories as if they were your own.

To my dearest nephew, Yanis,
From your first steps to the adventures that await, may this dedication remind you of the joy you bring and the love that surrounds you. Grow, explore, and know that your auntie is always here, cheering you on every step of the way. Your curious eyes and infectious giggles brighten even the darkest days.

To my amazing friend, Yasmine,
Your laughter and unwavering support have been my strength during moments of doubt and stress. Thank you for always lending me your ear and for being the amazing companion during the highs and lows of this journey. Your unwavering backing has been my rock through stormy periods.

To my dear Friends, Nihad , Narimene, Ines,
As I reflect on the past five years, I am filled with gratitude for your friendship. Thank you for your encouragement, laughter, and for being a constant source of joy. Thank you for your understanding, and for the countless memories we've made.

To my dedicated partner, Line Sarra,
your invaluable contributions, encouragement, and collaborative spirit have made this project a success. I am grateful for our teamwork. Thank you for your hardwork support, and understanding in every situation.

Manel Hamadou

Acknowledgment

In the name of Allah, the Most Beneficent and the Most Merciful, we begin by expressing our heartfelt gratitude to Allah for His blessings and strength that enabled us to complete this work.

We deeply appreciate our supervisor, **Dr. Elhocine BOUTELLAA**, whose invaluable guidance, consistent support, and expertise were crucial throughout this research journey. His feedback and encouragement helped us navigate challenges and achieve our goal, significantly shaping the quality and success of this study.

Furthermore, we proudly acknowledge our own dedication, motivation, and hard work in doing the research and putting together this project despite facing all the challenges throughout the year.

Our gratitude also extends to our family and friends for their belief in us and continuous encouragement. Additionally, we are grateful to all of the people who contributed to the effective completion of this project, their support and guidance were essential in its accomplishment.

Hamadou Manel, Benchemam Line Sarra

Abstract

Since its inception by Vaswani et al. [31] in 2017, the Transformer model has revolutionized various fields, including natural language processing, computer vision, audio data processing, etc. The aim of this project is to investigate the potential of Transformers for solar power production prediction. This task, though challenging, has very interesting practical applications, including grid management, proactive maintenance planning, fault detection, fault prediction, etc. Our main goal of this Master project is to design a transformer model and optimize its parameters for solar power production forecasting and fault prediction tasks. The key features of our proposed Transformer model include relying on a encoder-only architecture, integrating static positional encoding, and incorporating the Kolmogorov-Arnold Network as a feed-forward part. Experimental results indicate that our Transformer model achieves Outstanding performance in both anomaly detection and forecasting tasks. Specifically, the Transformer model excels in capturing long-range dependencies in time series data, leading to more accurate and reliable predictions. It significantly outperforms the LSTM model in several key metrics, demonstrating its superior ability to handle the complexities inherent in solar power production data. This superior performance underscores the transformative potential of the Transformer model in enhancing the accuracy and efficiency of solar power prediction, making it a vital tool for improving grid management, optimizing maintenance schedules, and advancing fault detection mechanisms.

key words: Transformer model, Solar power production prediction, Time series forecasting, Anomaly detection, Kolmogorov-Arnold Network.

List of Figures

List of Tables

Contents

List of Acronyms

ADAM Adaptive Moment Estimation

Adamax Adam with Infinity Norm

AI Artificial Intelligence

AMSGrad AMSGrad variant of ADAM

ANN Artificial Neural Network

API Application Programming Interface

ARIMA Auto Regressive Integrated Moving Average

BPNN Backpropagation Neural Networks

CMM Coarse Matching Module

CNN Convolutional Neural Network

DL Deep Learning

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

GAN Generative Adversarial Network

GDP Gross Domestic Product

GHI Global Horizontal Irradiance

GPUs Graphics Processing Units

GRU Gated Recurrent Units

KAN Kolmogorov Arnold Networks

LLM Large Language Model

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MLP Multilayer Perceptron

MSE Mean Squared Error

Nadam Nesterov-accelerated Adaptive Moment Estimation

NaN Not a Number

NLP Natural Language Processing

NWP Numerical Weather Prediction

PE Positional Encoding

PV Photovoltaic

RBF Radial Basis Function

ReLU Rectified Linear Unit

RMSprop Root Mean Square Propagation

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

TF Transformer

TN True Negative

TP True Positive

VAE Variational Autoencoder

Introduction

The integration of renewable energy sources into the global energy mix has become increasingly important due to growing concerns about climate change and the need for sustainable practices. Solar power, in particular, has emerged as a vital component of this transition, offering a clean and renewable source of energy. However, the intermittent nature of solar energy production requires accurate future prediction mechanisms to ensure the reliability and efficiency of solar power systems.

Monitoring solar energy systems rely on continuously collecting data from deployed sensors. The data form, consisting of time stamps and measured values, is commonly refereed to as time series. Time series forecasting has become a crucial tool in various domains, including economics, finance, engineering, and meteorology, driven by the need for accurate predictions and extended prediction horizons. Particularly, solar power forecasting is critical for ensuring the reliability and efficiency of power systems. However, traditional methods often struggle to accurately capture the complexities inherent in solar power production data, which will be discussed in more detail in Chapter 3. The Transformer model, which has revolutionized various fields including natural language processing, computer vision, and audio data processing, offers a promising approach to address this challenge and presents a promising avenue for enhancing the accuracy and efficiency of solar power prediction. The primary objective of this master project focuses on the development and optimization of a Transformer-based model for solar power forecasting and potentially anomaly prediction.

The proposed model relies on an advanced architecture that integrates static positional encoding and incorporates the Kolmogorov-Arnold Network as a feed-forward part. Through a comprehensive experimental evaluation, encompassing several metrics, for power forecasting and fault prediction tasks and comparative analysis against existing methods, this research aims to provide valuable insights into the transformative potential of Transformer models in revolutionizing solar power prediction to achieve precise forecasts, facilitate efficient energy production, grid management, and proactive maintenance strategies. By conducting detailed analysis and discussions on outcomes, this study aims also to clarify the intricacies of Transformer-driven solar power prediction models, presenting insights into their strengths, and potential pathways for further enhancement and real-world application. Ultimately, this study aims to contribute meaningful advancements to the field of solar power forecasting, addresses key challenges such as anomaly detection and long-term dependency modeling. supporting the broader integration of renewable energy sources and the development of more reliable and efficient energy systems.

To thoroughly explore the required core aspects, this report is organized into four chapters. Chapter 1 provides the theoretical foundations of time series analysis and forecasting, highlighting the importance of the topic and the various approaches used. Chapter 2 critically assesses prior studies, identifying areas for improvement and innovation specifically within solar power prediction, while mentioning the impact of Transformer models on time series forecasting, focusing specifically on their revolutionary application in solar power prediction. Chapter 3 presents our proposed methodology for designing the Transformer model, including the network architecture and training procedure. Chapter 4 delves into the experiments, results, and discussions regarding the optimization of our Transformer model and conducts a comparative analysis to show the outstanding results of our Transformer model over the LSTM model. Finally, we conclude our report by discussing the implications of our findings, highlighting the significance of our work, and outlining future research directions.

Chapter 1 Theoritical Background

1.1 Introduction

Time series analysis and forecasting are important concepts with compelling applications in a wide range of industries. They enable making predictions about future data points based on patterns and trends observed in the past. In this chapter, we will provide a comprehensive overview that is essential for understanding time series analysis and forecasting. We will cover fundamental concepts, examine applications in diverse fields, and emphasize the benefits of time series analysis. Additionally, we will discuss different approaches and methods utilized in time series forecasting, encompassing both traditional and contemporary techniques. Importantly, we will investigate recent innovations in this domain, particularly emphasizing Transformer models that have significantly contributed to time series forecasting, which forms the core focus of our report.

1.2 Definition

Time series data represents a sequence of data points recorded at regular intervals, offering a chronological perspective on a specific topic, such as stock prices, energy consumption, or weather patterns. To analyze such data effectively, it is essential to decompose the time series into its constituent components: trend, seasonality, and residuals. This decomposition process involves separating these elements to gain a deeper understanding of the data structure, facilitating the recognition of underlying patterns and informed forecasting [18]. In the following, these important concepts are outlined.

- **Trend:** Refers to the underlying direction in which the data is moving over the long term. This could be an upward or downward trajectory, indicating a gradual increase or decrease over time.

- **Seasonality:** Encompasses recurring patterns or cycles that occur at regular intervals, such as daily, monthly, or yearly fluctuations. These patterns are often tied to specific events or times of the year.

- **Residuals:** Represent the random variability or ”noise” in the data. These are the unpredictable elements that aren't explained by either the trend or seasonal components.

An example a time series decomposition of a company earnings per share is shown in Figure 1.1 . The top graph, labeled as observed, simply shows the time series as it was

recorded. The observed data is then split into trend, seasonal, and residuals.

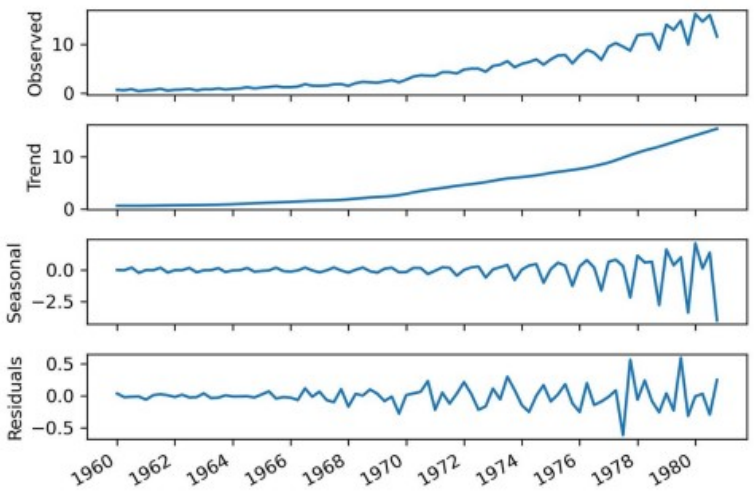


Figure 1.1: Decomposition of quarterly earnings of Johnson & Johnson from 1960 to 1980[18].

1.3 Applications of time series analysis

Time series analysis is a versatile tool with applications that span multiple industries, allowing for the examination of data trends and variations. This analytical approach offers valuable insights that inform decision-making, operational efficiency, and strategic planning. Below, we explore various fields significantly impacted by time series analysis.

Financial forecasting and economic analysis

Time series analysis is instrumental in financial forecasting and economic analysis, providing crucial insights that drive strategic decisions across various sectors, This includes:

- **Stock market and investment**: Time series analysis is important for forecasting stock prices, currency exchange rates, and commodity values, providing investors with data-driven insights for informed decision-making and risk mitigation.
- **Macroeconomic trends**: This approach helps economists analyze trends in key indicators like Gross Domestic Product (GDP), unemployment rates, and inflation aiding in policy development and business strategy formulation.
- **Demand and sales forecasting**: Anticipating future demand and sales trends is essential for optimizing business operations. Time series analysis provides valuable insights and predictive capabilities that businesses rely on for strategic decision-making, taking as an example:
  - **Demand planning:** Businesses utilize time series analysis to predict future demand for products and services. This enables more efficient inventory man-

agement and production planning, reducing excess stock and optimizing supply chains.

– **Sales projections:** Companies leverage time series analysis to forecast sales trends, guiding marketing initiatives, budget allocation, and resource distribution.

Signal processing

Time series analysis is employed in signal processing to extract meaningful information from noisy data. This is particularly useful in applications like speech recognition and biomedical signal analysis, enhancing the quality and reliability of these technologies.

Healthcare monitoring

Time series analysis is used in healthcare to monitor patient data, predict disease progression, and support clinical decision-making. This leads to improved patient care through early diagnosis and personalized treatment plans.

Predictive maintenance and quality control

Time series analysis plays a crucial role in manufacturing, ensuring product quality and consistency while detecting defects early to reduce waste and maintain compliance with industry standards. Additionally, it is essential for predictive maintenance, enabling industries to monitor equipment performance, anticipate failures, minimize downtime, cut maintenance costs, and enhance equipment reliability through a proactive approach.

Weather forecasting

Time series analysis is integral to weather forecasting, supporting predictions of temperature, precipitation, and extreme weather events. These forecasts are essential for disaster preparedness, agriculture planning, and public safety.

Energy generation and grid management

Relying on data insights to shape the future of energy, time series analysis is essential for ensuring the reliable integration of renewable energy. Various energy sectors depend on this analysis to optimize operations and planning. Key tasks include :

- **Energy forecasting:** Time series analysis is used to forecast energy production from various sources, including renewable energies. This aids in grid management, balancing supply and demand, and optimizing energy distribution.
- **Sustainability and environmental impact:** By analyzing time series data, policymakers and energy companies can evaluate the environmental impact of different energy sources, promoting sustainability and informing energy policies.

1.4 Time series analysis

In the realm of time series analysis, effectively detecting accruing events and predicting future values using machine learning tools is a challenging problem. Three common tasks: anomaly detection, anomaly prediction, and classification may be performed when processing time series data. These tasks are closely related and share similar goals, particularly in domains like renewable energy sector, where they play vital roles in optimizing energy generation and distribution. Using solar energy as an example provides a clear illustration of how these approaches distinguish themselves in practice.

- **Anomaly detection:** Identifies unexpected deviations from historical data, typically focusing on real-time or retrospective detection. The primary objective is to catch anomalies as they occur, allowing for quick response and correction. For example, in a solar panel farm, anomaly detection can be used to identify sudden drops in energy output due to panel malfunctions or shading, enabling prompt maintenance and minimizing energy losses.

- **Anomaly forecasting:** Involves predicting future anomalies based on past patterns. This approach goes beyond detection to anticipate potential irregularities before they occur, allowing for proactive risk mitigation and resource planning. This is particularly useful in contexts where anticipating disruptions can significantly reduce operational risks. For instance, by analyzing historical weather patterns and energy output data, anomaly forecasting can predict potential energy shortfalls due to weather conditions, enabling solar farms to schedule backup power sources or adjust energy storage accordingly.

- **Classification:** Classification in time series analysis refers to the process of categorizing time-dependent data into distinct groups or classes based on observed patterns and features. This method utilizes supervised learning algorithms to extract meaningful features, such as trends or periodic behaviors, and train models that can classify new data points into predefined categories. For example, classification can be used to categorize solar energy output data into different patterns, such as peak hours, off-peak hours, or periods of high variability. This classification can help optimize energy storage and distribution strategies, ensuring a more efficient and reliable energy supply.

These approaches share common goals and can be applied individually or in conjunction with each other across various fields to comprehensively understand unusual patterns and outliers. Integrating anomaly detection, anomaly forecasting, and classification enables researchers and practitioners to develop more effective strategies tailored to specific challenges, thereby enhancing the reliability and efficiency of systems in diverse domains.

1.5 Time series forecasting

Time series forecasting is a sub-field of statistical analysis that involves predicting future data points based on a sequence of historical data, trends and currently available information [18] [26]. It is an essential tool for understanding trends, patterns, and dependencies in time series data. Initiating a forecasting project involves following specific protocols, including

setting goals, determining forecast requirements, and progressing through various stages, as shown in Figure 1.2.

Time series forecasting has a wide range of practical applications across diverse domains. In particular, the solar power generation industry extensively utilizes time series forecasting to make precise predictions about future power output, as we will address in this report. This capability aids solar power plants and grid operators in effectively managing equipment and detecting faulty components. By harnessing the power of time series forecasting, solar power systems can operate efficiently and ensure maximum output [22].

When using time series forecasting, it is crucial to consider challenges such as noise and outliers, seasonality, nonstationarity, data length, and overfitting. These challenges can be addressed through careful data preparation, including cleaning, removing outliers, filling in missing values, transforming the data to make it stationary, and splitting the data into training and testing sets.

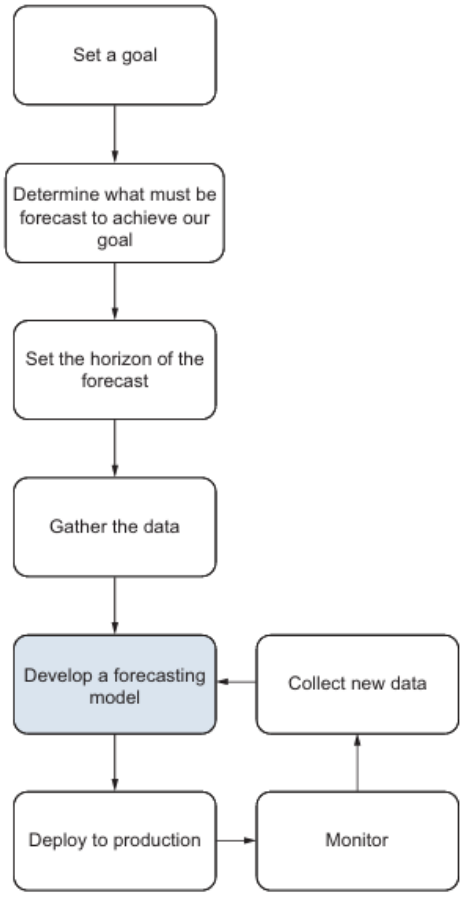


Figure 1.2: Time series forecasting workflow.

1.5.1 Forecasting versus regression

Time series forecasting presents notable distinctions compared to standard regression tasks. The temporal sequence of data holds paramount importance, maintaining its chronological order throughout the modeling process to predict future values based on past observations. Time series forecasting primarily relies on the inherent patterns within the time series data itself, employing techniques like moving average or autoregressive models.

These fundamental approaches facilitate the exploration of seasonal patterns and trends, underscoring the unique challenges and strategies within this specialized forecasting domain [18].

In contrast, traditional regression tasks focuses on understanding and quantifying the relationships between dependent and independent variables, using various models such as linear or logistic regression, and is widely applied in fields like economics and health research. While forecasting is primarily about future prediction, regression is about understanding the relationships between variables [15].

1.5.2 Time series forecasting techniques

Time series forecasting has evolved significantly, incorporating a range of methods from traditional approaches like Auto Regressive Integrated Moving Average (ARIMA) and Prophet to advanced deep learning techniques such as Recurrent Neural Networks (RNNs) with variants such as long-short-term memory (LSTM) and gated recurrent units (GRUs). This section provides an overview of the mentioned methods and introduces Transformer neural networks, which have played a crucial role in time-series forecasting.

1.5.2.1 Traditional techniques

This section outlines two famous traditional methods commonly used for time series forecasting tasks. These methods have been established as the state of the art for a while and greatly contributed to advancing time series analysis.

1. **ARIMA**

   ARIMA is one of the traditional statistical methods that has been extensively used in time series forecasting. It is an extension of the ARIMA model, integrating the differencing component to make it suitable for non-stationary time series data by transforming it into stationary data. ARIMA combines autoregression, differencing, and moving average concepts to model the dynamic behavior of a time series [8].

   The ARIMA equation is a regression type equation in which the independent variables are lags of the dependent variable and/or lags of the forecast errors [19]. The equation of the ARIMA model is given as:

$$y'(t) = c + \phi_1 \cdot y'(t-1) + \ldots + \phi_p \cdot y'(t-p) + \theta_1 \cdot \varepsilon(t-1) + \ldots + \theta_q \cdot \varepsilon(t-q) + \varepsilon_t \quad (1.1)$$

   where:

   - y_t is the actual value at time t,
   - c is a constant term,
   - $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive (AR) coefficients,
   - $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average (MA) coefficients,

- ε_t is the error term at time t,
- p is the order of the autoregressive part,
- q is the order of the moving average part.

2. **Prophet**

Prophet is an open-source forecasting tool developed by Facebook, designed specifically for forecasting time series data that exhibits strong seasonal effects and has several seasons of historical data. It excels in handling missing data, trend shifts, and adjusts automatically for holidays affecting the data. Implemented in both Python and R, Prophet is accessible to a wide range of users. Below, we explore the foundational principles of Prophet and its core components:

- **Additive Model Components**: Prophet operates on an additive model where the time series data $y(t)$ is decomposed into three main components:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{1.2}$$

where:

  - $g(t)$: Trend component capturing non-periodic changes in the time series.
  - $s(t)$: Seasonal component modeling periodic fluctuations.
  - $h(t)$: Holiday effects, accounting for significant deviations caused by user-specified events.
  - ϵ_t: Error term handling unusual changes not explained by the model.

- **Trend Component**: Prophet supports two trend models:
  - **Piecewise Linear**: Represented as

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma) \tag{1.3}$$

    where k is the initial growth rate, δ represents the slope change points, m is the offset parameter, and $a(t)$ are the adjustment factors.

  - **Logistic Growth**: Given by

$$g(t) = \frac{C}{1 + \exp(-k(t - m))} \tag{1.4}$$

    where C is the carrying capacity, k is the growth rate, and m is the time of maximum growth rate.

- **Seasonality Component**: Prophet models seasonality using Fourier series, effectively capturing periodic changes. For instance, yearly seasonality is modeled as:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \tag{1.5}$$

where P is the period (e.g., 365.25 for yearly data), and a_n, b_n are Fourier coefficients.

- **Holiday Effects**: Prophet incorporates holiday effects by allowing users to specify holidays and their impacts on the time series. These effects can be modeled as additional regressors or predefined holidays with specific effects.

  By leveraging these principles and functionalities, Prophet emerges as a robust tool for time series forecasting, applicable across various domains from business analytics to academic research.

1.5.2.2 Deep learning techniques

This section explores the evolution of deep learning methods and highlights prominent techniques widely adopted for time series forecasting tasks. Since the Transformer model is a main constituent of our work it will be presented in a separate section, though it falls under the present category.

1. **CNN**

   A convolutional neural network (CNN) is a deep learning architecture that utilizes the convolution operation to create a reduced set of features. This operation helps regularize the network, prevent overfitting, and effectively filter inputs.

   Convolution is performed with a kernel, which is trained during model fitting, and its stride determines how many steps it shifts at each convolution step. Originally designed for image processing, CNNs have been adapted for sequential data by applying convolutions across the temporal dimension, using 1D convolution for tasks like time series forecasting. This adaptation enables CNNs to capture patterns and dependencies within time series data, making them valuable in various domains [40]. Figure 1.3 shows the overall architecture of a convolutional neural network.

   - **Convolutional layers:** In a CNN for time series, the input data is treated as a 1D signal, with each time step as a feature dimension. These layers consist of filters (kernels) that slide over the input data to extract local patterns or features. By applying convolutions across the temporal dimension, the model captures patterns and relationships within the time series data [24].

   - **Pooling layers:** Following the convolutional layers, pooling layers downsample the feature maps, reducing their spatial dimensions while retaining important features. Common pooling operations include max pooling and average pooling, which respectively take the maximum or average value within each pooling window [24].

2. **LSTM**

   was first proposed in 1997 as a type of Recurrent Neural Network (RNN) that aims to model time series and their long-term dependencies more accurately than traditional RNNs and serves as a useful tool in time series forecasting [17, 21]. LSTM solves the problems of gradient explosion and vanishing gradient by adding memory cells

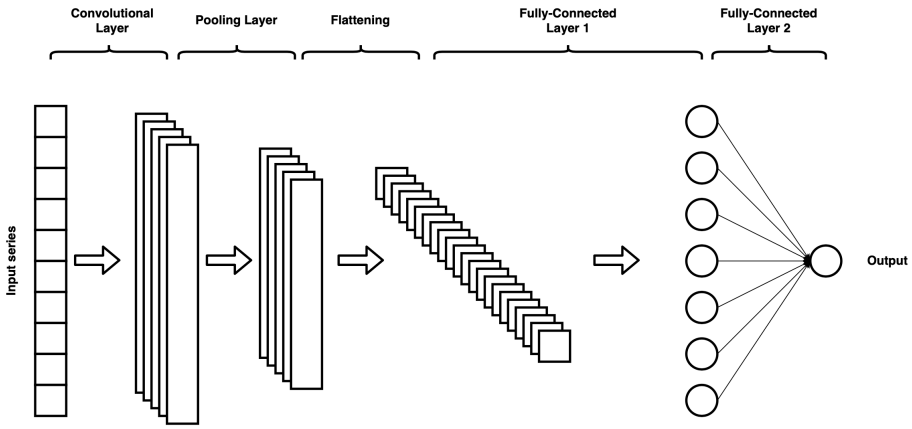


Figure 1.3: CNN architecture overview [24].

[9], which is a common issue in traditional RNNs when dealing with long sequences of data.

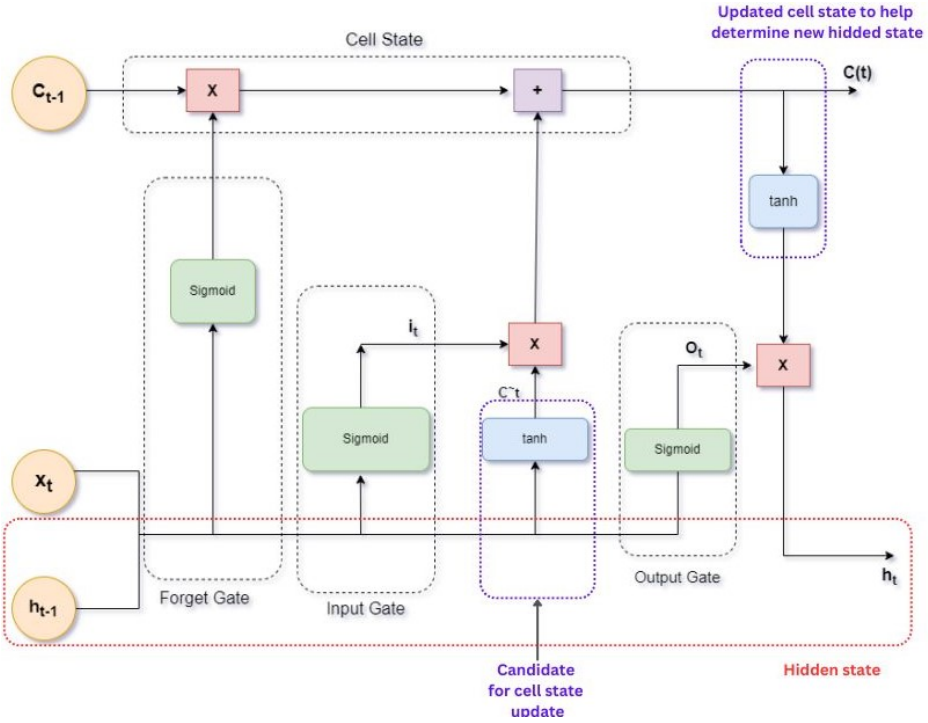


Figure 1.4: LSTM architecture.

The standard LSTM architecture consists of three primary components: the input layer, the recurrent LSTM layer, and the output layer. The input layer is followed by the LSTM layer, which includes three types of gates: the cell input unit gate, the output gate, and the forget gate. These gates form recurrent connections, with the direction of flow being from the output to the input. The output layer is then connected to the output units. This structure is illustrated in Figure 1.4 which was inspired from [30]. It can be seen that the memory cell is the basic unit of LSTM [9].

The LSTM consists of three gates that play distinct roles:

- **The forget gate:** determines which information from previous steps remains important.
- **The input gate :** decides which information from the current step is significant.
- **The output gate :** determines the information that is propagated to the next element in the sequence or contributes to the final output layer [18].

3. **GRU**

GRU is a type of recurrent neural network and was introduced in [2], as a simpler alternative to LSTM networks. Its main idea is similar to that of LSTMs, as it similarly uses gating mechanism to control how much to remember from previous states.

However, it is a little bit simpler and has fewer parameters. It does not have a dedicated cell state and introduces the reset and update gates.

- **The reset gate :** determines how much of the previous hidden state should be forgotten[2].
- **the update gate :** determines how much of the new input should be used to update the hidden state.

Generally, it is simpler than the LSTM and is hence faster to train and less prone to overfitting and its output is calculated on the basis of the updated hidden state.(e.g., when used with time series). Figure 1.5 gives an overview of the GRU cell.

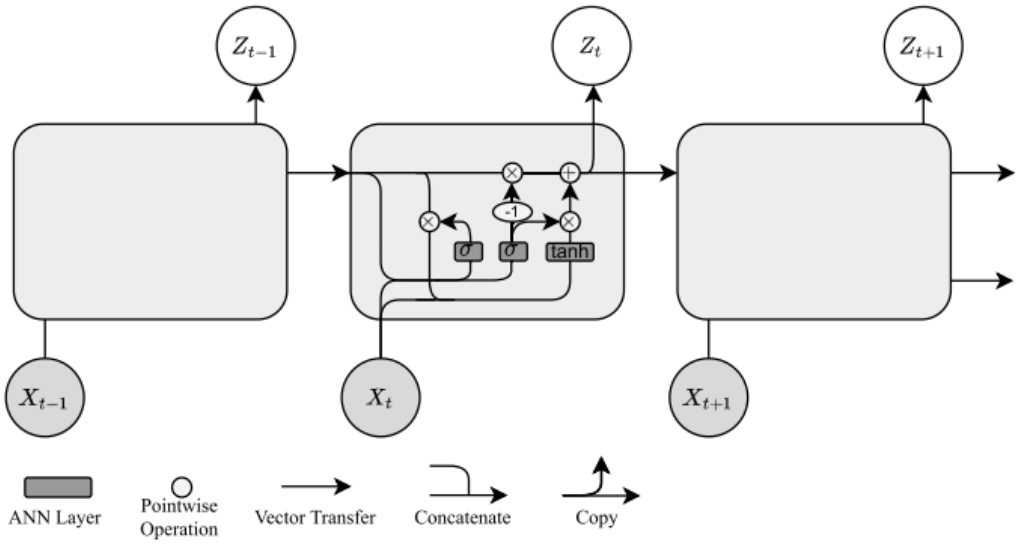


Figure 1.5: Overview of a GRU cell unrolled in time [26].

In certain cases, crucial information might appear early in a sequence, making it difficult for traditional methods to capture effectively, especially in sequence prediction problems. To address this challenge, the Transformer deep learning model was introduced. The Transformer is a state-of-the-art model, and we will focus on it in detail in a different section.

1.6 Introduction to Transformers

The Transformer model is a type of deep learning neural networks,it was introduced by Google in 2017. It was specifically developed to use attention mechanisms for handling sequential data, aiming to solve the challenges of sequence learning in natural language processing tasks like machine translation [31].

The core concept of the Transformer model is to facilitate the conversion of an input sequence from one domain to an output sequence from another domain. For instance, it allows us to train a robot to translate English sentences into French. Similarly, when we view a segment of a time series as a sentence in one language and the subsequent segment as a sentence in another language, the task of multi-step time series forecasting can be seen as a sequence learning problem as well. Hence, the Transformer model offers a viable approach to tackle time series forecasting in the field of data analysis. As mentioned in [36], the Transformer model has been adapted in numerous ways to perform time series forecasting tasks, as we will see in the upcoming chapter.

1.6.1 Attention mechanism

It is also known as intra-attention, is a system for weight distribution that enables the network to allocate importance to various words or tokens in a given set irrespective of their placement in the input or output sequence. This is achieved by computing attention scores for each word pair in the input sequence[15]. The Transformer is the first model that can solely depend on self-attention without the need for sequence-aligned RNNs or convolutional layers.

In self-attention, each input element (e.g., word, token, or time step) is associated with query vector, key vector, and value vector. These vectors are derived from the input embedding via linear transformations, which are tuned through training. The attention score between a Query vector and a Key vector measures the relevance or importance of the Key with respect to the Query. This score is then used to weight the corresponding Value vector.

- **Query (Q):** refers to the specific element for which attention scores need to be calculated. Typically, the query is compared with all other elements in the sequence.
- **Key (K):** contains information that is utilized to assess the relevance of other elements to the query. The attention mechanism analyzes the similarity between each key and the query.
- **Value (V):** holds the actual information that contributes to the output.

Attention scores quantify the degree of attention or focus that one element should pay to another during computation and it is calculated, using the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.6}$$

where:

Q is the Query matrix.

QK^T is the dot product between the query and key vectors, measuring similarity.

V is the values weighted by the attention scores to produce the output.

$\sqrt{d_k}$ is kth scaling factor based on the dimensionality of the key vectors.

Here, the softmax forms the convex combination weights for the values in V , allowing us to compress the matrix V into a smaller representative embedding for simplified inference in the downstream neural network operations.

1.6.2 Transformer architecture

In this section, we will go through the architecture shown in figure 1.6, which represents the overall architecture of the standard Transformer model. It is based on The encoder-decoder architecture. All encoders have the same architecture and decoders share the same property. We will break down each component and explain its role in the overall architecture.

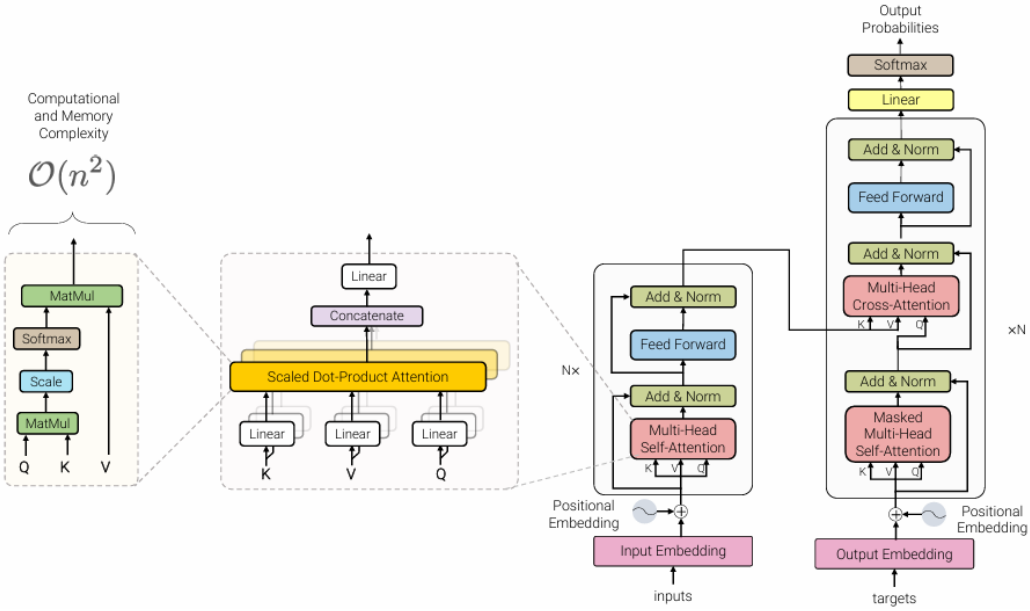


Figure 1.6: Standard architecture of a Transformer [28].

1. **Positional encoding** : To use the sequence order in the model, it is essential to incorporate information regarding the relative or absolute position of tokens within the sequence. This is achieved by introducing *"positional encodings"* to the input embeddings at the encoder and decoder stacks. These positional encodings have the same dimension as the embeddings, allowing them to be added together. There are various options available for implementing positional encodings [36].

2. **Encoder** : it processes the input sequence and generates a series of contextualized representations for each token in the sequence. These representations capture information about the token's relationships with other tokens in the input. The encoder

is composed of a series of N identical encoder blocks connected in sequence. Each encoder block, is made up of six layers, with each layer containing two sub-layers. The first sub-layer is responsible for multi-head self-attention, whereas the second sub-layer applies a feed-forward network [31] [8].

- **Multi-head self-attention layer** : This layer measures the relevance of each element in relation to all other elements in the sequence. It applies the self-attention mechanism multiple times in parallel independently referred to as "heads", using different learned linear projections to generate the queries, keys, and values.

  The outputs from each attention head are concatenated and linearly transformed to produce the final multi-head attention layer output, enabling the model to capture diverse relationships and aspects of the input sequence simultaneously, enhancing flexibility and information understanding.

Let h represent the number of heads in the multi-head self-attention mechanism. Given an input X, we have:

$$Q_i = X \cdot (W_i)^Q \tag{1.7}$$
$$K_i = X \cdot (W_i)^Q \qquad \text{for } i = 1, \ldots, h \tag{1.8}$$
$$V_i = X \cdot (W_i)^V \tag{1.9}$$

where $W_i^Q \in \mathbb{R}^{(d_{input} \times d_{head})}, W_i^K \in \mathbb{R}^{(d_{input} \times d_{head})}, W_i^V \in \mathbb{R}^{(d_{input} \times d_{head})}$

Then, each head is an attention score matrix:

$\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$

$$\text{Self-MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) \cdot W_O \tag{1.10}$$
$$= [\text{head}_1, \ldots, \text{head}_h] \cdot W_O \tag{1.11}$$

- **Fully connected feed-forward network of two layers:** This network operates on each position of the input independently and uniformly. This involves two linear transformations separated by a ReLU activation. The output of this network:

$$\text{FFN}(x) = \text{ReLU}(W_1 x + b_1) W_2 + b_2 \tag{1.12}$$

where W_1, b_1 and W_2, b_2 are respectively the weights and bias parameters of two layers. Although the linear transformations remain consistent across positions, they utilize distinct parameters as we move from one layer to another.

The two sub-layers are enveloped in a residual connection and subsequently subjected to layer normalization (applied after each sub-layer). Consequently, the output of

each sub-layer is obtained by adding the input to the sub-layer and then normalizing it through layer normalization and passing it to the decoder [31].

3. **Decoder:** The Transformer model's decoder generates an output sequence based on the encoded input sequence. Operating in an autoregressive manner, it generates one token at a time while considering both previously generated tokens and encoded input information. This mechanism is crucial for tasks such as language translation, text summarization, and text generation.

   Similar to the encoder, the decoder consists of N = 6 identical layers, each with residual connections around its sub-layers. In addition to the two sub-layers present in each encoder layer, the decoder inserts a third sub-layer that performs multi-head attention over the output of the encoder stack[31].

   Each of these sub-layers self-attention, encoder-decoder attention, and feedforward features a residual skip connection followed by layer normalization. This overall structure and process enable the Transformer decoder to generate accurate, contextually appropriate, and coherent output sequences for various applications, ensuring effective performance in language processing tasks like translation and summarization.

   The structure of each decoder layer is as follows: the first decoder in the stack receives its input from the output embedding and positional encoding, while subsequent decoders receive input from the previous decoder. Each decoder layer includes a masked multi-head self-attention mechanism, which differs from the encoder's multi-head self-attention by only attending to earlier positions in the sequence.

   - **Masked Multi-head Self-attention Mechanism:** During training, this ensures that the model predicts each token based only on previously generated tokens without accessing future information. By masking certain positions in the decoder input, the model can focus solely on the relevant information needed for each prediction step, enhancing efficiency.

   Furthermore, the decoder incorporates a second multi-head attention layer known as the encoder-decoder attention layer. This layer combines inputs from two sources: the self-attention layer below it and the output of the encoder stack.

   - **Encoder-decoder Attention Mechanism:** After passing through layer normalization, the encoded representation is fed to the query parameter in the encoder-decoder attention mechanism. This layer combines information from the decoder's self-attention with the output of the encoder stack, computing the interaction between each target word and each input word. This process captures the influence of attention scores from the input sequence, helping the model to learn complex relationships between input and output sequences. The final encoder's output in the stack is fed into the value and key parameters for encoder-decoder attention. The encoder-decoder attention mechanism thus receives a representation of both the target sequence (from the decoder self-attention) and the input sequence (from the encoder stack). It produces a representation with the attention scores for each target sequence word that captures the influence of the

attention scores from the input sequence as well.

The output of the encoder-decoder attention layer is processed by a feedforward layer, and its output is then passed upwards to the next decoder layer.

1.6.3 Tranformers versus RNNs

The key distinction between the attention mechanism and traditional RNN or LSTM models is that the attention mechanism directly prioritizes specific segments of the sequence instead of treating all segments equally based on their order. This unique approach enables the model to capture information from early positions in the sequence more effectively. Consequently, the attention mechanism enhances the model's comprehension of the sequence's contextual variations [8].

Transformers enable significantly faster detection in comparison to recurrent methods by leveraging parallelization during inference on Graphics processing units (GPUs) [6]. In addition, they offer the advantage of accurately encoding large sequences, with training and inference times that are almost independent of the sequence length. Therefore, Transformer are used to enhance the temporal context information provided to an anomaly detector, without imposing significant computational burdens[29].

1.7 Conclusion

In this chapter, we explored the fundamental concepts behind the essence of our project. We introduced time series analysis and its wide range of applications. We discussed various methods for time series forecasting, emphasizing their specific challenges. We finished by introducing the Transformer model and outlined its power and capabilities. This knowledge forms a solid foundation for the next chapter, where we will delve into the existing research and examine how it informs our own approach.

Chapter 2 Related Work

2.1 Introduction

Transformers, revolutionizing the landscape of data analysis, excel in capturing intricate long-range dependencies through dynamic self-attention mechanisms. This transformative capability extends seamlessly into the energy sector, where Transformers redefine solar power forecasting. By accurately modeling the complexities of solar irradiance data, they empower robust grid management, optimize energy distribution, and ensure the reliability of renewable energy systems. This pivotal role underscores their pivotal role in advancing sustainable energy solutions and shaping the future of renewable energy integration.

This chapter provides a thorough review of Transformer models' impact on time series forecasting, focusing specifically on their revolutionary application in solar power prediction. It begins by exploring various Transformer architectures designed for forecasting. Then, it delves into the solar energy sector, discussing traditional solar energy prediction methods while highlighting their limitations. The report then shifts its attention to the application of Transformer-based models in solar power forecasting, examining key studies and the emergence of hybrid models that combine multiple architectures. A comparative analysis with conventional techniques in the solar energy domain elucidates the key findings of each approach. Finally, the chapter outlines future research directions and emphasizes the critical role of advancing AI integration, particularly in leveraging the dual capabilities of time series anomaly detection and forecasting to enhance the accuracy of solar power predictions

2.2 Transformers for time series forecasting

Transformer-based models have revolutionized various fields by providing powerful tools to capture complex dependencies in data. In time series forecasting, their ability to model intricate temporal relationships and handle diverse data contexts makes them particularly effective. This section explores three prominent Transformer architectures that have significantly impacted the field of time series forecasting: the Temporal Fusion Transformer (TFT), the Enformer, and TimesFM.

Temporal Fusion Transformer (TFT)

Introduced by Lim et al. (2019) [12], the Temporal Fusion Transformer (TFT) represents a significant advancement in time series forecasting through its innovative encoder-decoder

architecture. This model integrates Transformer layers specifically designed to handle both static and dynamic covariates, enhancing interpretability and performance in predicting future values over multiple time steps.

The TFT architecture begins with the encoder, which processes input data. This includes static covariates that remain constant over time, such as geographical location or categorical variables like the day of the week, and dynamic covariates that change over time, such as weather conditions, economic indicators, or past values from the time series. By incorporating both types of covariates, TFT gains a comprehensive view of the contextual factors influencing the time series data, allowing it to effectively capture intricate temporal patterns and dependencies.

Central to TFT's functionality is the use of Transformer mechanisms, particularly multi-head attention. Multi-head attention allows the model to focus on different aspects of the time series data simultaneously. This means that the model can identify significant patterns across various time scales, which is crucial for robust forecasting over extended horizons. In addition to multi-head attention, TFT employs a series of specialized layers and modules designed to handle the unique challenges of time series forecasting. For example, the gating mechanism in TFT helps the model select which parts of the data are most relevant for making predictions, thereby improving the interpretability of the model's outputs. The model also incorporates a temporal fusion decoder, which combines information from multiple time steps and covariates to generate final forecasts. This decoder phase is crucial for synthesizing the learned representations from the encoder phase into accurate predictions for future time steps.

One of the standout features of TFT is its ability to handle missing data effectively. In many real-world applications, data can be incomplete or noisy. TFT's robust architecture ensures that it can still make reliable predictions even when some data points are missing. This capability is particularly valuable in practical scenarios across various domains, such as energy, traffic management, and finance. In energy forecasting, for example, TFT has been successfully used to predict electricity demand and renewable energy generation, aiding in optimized grid operations and resource planning. In traffic prediction, the model helps forecast traffic flow and congestion patterns, contributing to the development of smarter urban transportation systems. Financial applications benefit from TFT's predictive capabilities in forecasting stock prices and economic indicators, providing decision-makers with timely insights for strategic planning and risk management.

Enformer

Traditional Transformer models face significant challenges when applied directly to time-series data, primarily due to high computational complexity and redundancy issues. To address these limitations, Wang et al. [34] designed "Enformer: Encoder-Based Sparse Periodic Self-Attention Time-Series Forecasting," a lightweight Transformer model that discards the traditional encoder-decoder structure in favor of a streamlined encoder-only architecture.

The Enformer architecture is built around an efficient encoder that processes the input sequence to capture essential temporal dependencies. A key feature of this design is the Coarse Matching Module (CMM), which generates multi-scale outputs from the input

sequence, enabling the model to capture dependencies across different time granularities. This multi-scale representation is crucial for understanding the hierarchical nature of time series data, where patterns may emerge at various levels of granularity. Additionally, the Enformer introduces a novel sparse periodic self-attention mechanism. Unlike the dense attention mechanisms in standard Transformers, this sparse periodic attention focuses on the periodic dependencies within the data, significantly reducing computational load while preserving essential temporal relationships. By concentrating on periodic patterns, the model efficiently captures long-range dependencies without the overhead of processing every possible interaction within the data. Empirical evaluations demonstrated that this innovative architecture improved forecasting accuracy by an average of 7.1% across multiple datasets and increased prediction time efficiency by 44.1%, making it more suitable for practical applications in time-series forecasting. These findings underscore that tailored modifications to the Transformer architecture can effectively optimize time-series forecasting, balancing computational demands with predictive performance.

The Enformer model is particularly useful in various fields that rely on time-series data. In the energy sector, it can be employed for load forecasting and predicting renewable energy generation, aiding in efficient grid management and resource allocation. In finance, the Enformer helps in forecasting stock prices and market trends, providing crucial insights for investment strategies and risk management. Additionally, in healthcare, the model can be used for monitoring patient health metrics and predicting disease outbreaks by analyzing historical health data. Its ability to handle long-range dependencies and computational efficiency makes it a valuable tool for diverse applications in time-series forecasting.

TimesFM

Another innovative approach is presented in the paper "A Decoder-Only Foundation Model for Time-Series Forecasting" [33], which introduces TimesFM, a model inspired by large language models (LLMs) and designed for time-series forecasting. TimesFM employs a decoder-only architecture, pre-trained on an extensive corpus of 100 billion real-world data points. This pre-training enables TimesFM to perform zero-shot forecasting, delivering accurate predictions on new datasets without additional training.

The TimesFM architecture utilizes a patching technique, which divides the time-series data into smaller patches. These patches are processed through stacked Transformer layers with self-attention mechanisms, effectively capturing dependencies across different temporal scales. This approach allows the model to understand and leverage both short-term and long-term dependencies within the time series data. A key component of TimesFM is the multilayer perceptron (MLP) block with residual connections. This block transforms the processed patches into tokens used for prediction, enabling the model to forecast longer future segments from shorter input patches. The residual connections help maintain the integrity of the information passed through the layers, improving the model's overall performance and stability.

TimesFM is versatile and useful across various domains. In the retail industry, it can predict sales trends and inventory needs, enhancing supply chain management and optimizing stock levels. For climate science, it can forecast weather patterns and climate changes, supporting environmental planning and disaster preparedness. In the healthcare sector, it can monitor patient health metrics and predict disease outbreaks by analyzing historical health

data. The model's flexibility in adjusting context length and prediction horizon makes it especially effective for handling diverse forecasting scenarios, demonstrating the power of pre-training a decoder-only model for versatile and precise time-series predictions.

2.3 Solar power forecasting

Solar power stands at the forefront of the global shift towards sustainable green energy solutions, offering abundant, clean energy with minimal environmental impact. The effective utilization of solar energy critically depends on accurate forecasting of solar irradiance, which directly influences photovoltaic (PV) module output. However, this forecasting is inherently challenging due to the dynamic and multifaceted nature of solar data, influenced by factors such as weather patterns, geographical variations, and technological configurations. Inaccurate predictions not only impact the efficiency of energy generation but also pose risks to grid stability and economic viability.

Over the years, solar forecasting techniques have undergone significant evolution. Initially relying on traditional statistical methods like time series analysis and numerical weather prediction, the field has seen a paradigm shift towards the adoption of deep learning solutions, including revolutionary approaches such as Transformer-based models and hybrid methodologies. These advancements have transformed solar forecasting by harnessing advanced machine learning techniques capable of capturing intricate temporal and spatial dependencies within solar data. This section explores and evaluates these progressions, highlighting their transformative impact on enhancing the reliability and efficiency of solar energy systems.

2.3.1 Traditional methods

The historical progression of traditional forecasting approaches underscores their pivotal role and enduring impact on the field of solar energy research. Traditional methods, such as time series analysis, regression models, and Numerical Weather Prediction (NWP), were the cornerstone for understanding and predicting solar irradiance in the early stages of solar power forecasting. These approaches laid the groundwork for subsequent advancements in forecasting accuracy and reliability by establishing fundamental principles and techniques. Despite their simplicity compared to modern deep learning methods, traditional approaches have shaped the field by providing a foundation for understanding the complex relationships between weather patterns, atmospheric conditions, and solar irradiance.

- **Time Series Analysis Techniques:** Time series analysis, such as ARIMA models and exponential smoothing, has been foundational for short-term solar forecasting [1]. These techniques focus on capturing temporal patterns and dependencies within solar irradiance data, providing valuable insights into immediate fluctuations.

- **Regression Models:** Regression models establish relationships between solar irradiance and meteorological variables like cloud cover, temperature, and humidity [38]. By quantifying these correlations, regression methods contribute to understanding how environmental factors impact solar energy production.

- **Numerical Weather Prediction (NWP):** NWP models represent a significant ad-

vancement in medium to long-range solar forecasting. These models simulate atmospheric processes using complex mathematical equations, including the Navier-Stokes equations, within three-dimensional grids that simulate the Earth's atmosphere [14]. By assimilating data from ground stations, satellites, and weather balloons, NWP models generate detailed forecasts of weather variables critical for predicting solar radiation.

Despite their usefulness, traditional statistical methods have several limitations. They often struggle to capture the complex non-linear relationships and high-dimensional interactions inherent in solar irradiance data. Additionally, these methods are sensitive to data quality and may not perform well in the presence of missing or erroneous observations. Furthermore, traditional methods typically rely on a limited set of input variables, failing to exploit the wealth of data available from various sources, such as satellite imagery, sensor networks, and historical records.

2.3.2 Deep learning methods

Research conducted by Husein et al. [7] demonstrates that day-ahead solar irradiance forecasting can lead to substantial annual energy cost savings in commercial building microgrid operations. This underscores the critical importance of selecting tailored forecasting methods for solar power applications, thereby reducing dependence on nonrenewable energy sources. Recent studies increasingly advocate for deep learning approaches in solar power time series forecasting, citing their ability to significantly enhance both accuracy and operational efficiency.

- **Shallow Neural Networks**: One of the earliest applications of AI in solar forecasting involved shallow neural networks, such as multilayer perceptrons (MLPs) and radial basis function (RBF) networks [23]. These models take meteorological variables and historical solar irradiance data as inputs and predict future solar power generation. While shallow neural networks have shown promising results, they are limited in their ability to capture long-range dependencies and handle high-dimensional data.

- **Convolutional Neural Networks (CNNs)**: convolutional Neural Networks (CNN) plays a crucial role in spatial-temporal modeling ,which refers to the analysis of how phenomena change over both space and time, of solar irradiance by leveraging satellite imagery and sensor network data [5]. CNNs excel in extracting intricate local features and capturing spatial relationships, making them particularly effective for modeling how cloud patterns affect solar irradiance.

  Suresh et al. [27] proposed an innovative CNN-based approach that includes various architectures: standard CNNs, CNNs with multiple heads, and CNN–LSTM hybrids. These architectures are integrated with sliding window data-level approaches and advanced data preprocessing techniques, enhancing their ability to generate precise forecasts. By linking the output of solar panels with input parameters such as sunlight intensity, module temperature, ambient air temperature, and wind speed, this approach comprehensively models the complex interactions influencing solar power generation.

- **Recurrent Neural Networks (RNNs)** Recurrent Neural Networks (RNNs), includ-

ing gated recurrent units (GRUs) and particularly long short-term memory (LSTMs), are widely employed for time series forecasting in solar power prediction. LSTMs excel in capturing intricate, nonlinear relationships between weather variables and photovoltaic (PV) power output, making them highly suitable for handling the complex dependencies inherent in sequential solar data[11].

LSTMs are especially adept at modeling the long-range dependencies present in sequential solar data, effectively capturing the dynamic nature of solar irradiance. Their strong performance in short-term solar power forecasting is crucial for ensuring grid stability and effective cost management. Recent advancements include the exploration of transfer learning techniques using stacked LSTM architectures, addressing data scarcity issues, particularly for newly installed plants. This approach enables models trained on one location to generalize effectively to another with limited historical data, mitigating the inherent data-hungry nature of deep learning models [37].

Studies have consistently demonstrated that LSTM models achieve lower root mean squared error (RMSE) compared to traditional physical models. Additionally, the use of Rectified Linear Unit (ReLU) activation functions within LSTMs helps minimize prediction errors during adverse weather conditions such as snowfall. Elsaraiti et al. [3] proposed a deep learning method using LSTM models to accurately estimate daily solar power levels in Halifax, Nova Scotia, Canada, utilizing solar power data from January 1 to December 31, 2017. Their work underscores the effectiveness of LSTM models in capturing temporal dependencies and outperforming traditional methods in various solar forecasting tasks.

Neural Basis Expansion Analysis Time Series (N-BEATS) An impactful innovationis the N-BEATS, developed by Oreshkin et al. (2020)[16], is structured around a layered architecture designed for robust time series forecasting, particularly effective in applications like solar power prediction. the proposed model is a deep neural architecture based on backward and forward residual links and a very deep stack of fully-connected layers. At its core, N-BEATS features stacked layers, each comprising trend and seasonality blocks. The trend blocks capture long-term patterns using feed-forward neural networks, while the seasonality blocks model periodic components such as daily or weekly variations. These blocks utilize neural basis functions instead of traditional methods like Fourier transforms, allowing N-BEATS to adaptively learn complex temporal relationships within the data. Skip connections between layers facilitate the propagation of information, enhancing gradient flow and preserving crucial data features across the network. At the top, an output layer integrates predictions from all blocks to produce the final forecast, ensuring comprehensive coverage of both trend and seasonal aspects. During training, N-BEATS optimizes its parameters through backpropagation, aligning its predictions with actual data values. This approach not only enhances performance in traditional forecasting tasks like electricity load prediction but also demonstrates significant potential in advancing the accuracy and reliability of solar power generation forecasts. The proposed architecture was tested on several well-known datasets, including M3, M4, and TOURISM competition datasets containing time series from diverse domains.

Transformer-based models in solar power prediction

The sun, a seemingly constant source of energy, presents a surprisingly complex challenge when it comes to predicting its power output. Forecasting solar power production is a complex task due to the intricate interplay of various factors. Weather patterns, fleeting clouds, and even subtle shading effects all dance together, creating a dynamic and non-linear puzzle. Traditional forecasting techniques, and even some deep learning methods, struggle to capture the intricate relationships and variations present in solar power data, which is inherently nonstationary. However, the Transformer model emerges as a powerful solution, offering a unique self-attention mechanism that allows it to identify global dependencies and connections within the data sequence. This ability to capture long-range dependencies and complex variations makes the Transformer model particularly effective in modeling solar power time series data, leading to more accurate forecasts. In this section, we will delve deeper into existing models and their architectures, exploring how the Transformer surpasses them in tackling this crucial challenge.

Deep learning global horizontal irradiance prediction (DL-GHIP) model

An initial deployment of Transformer models in solar power prediction was the work of Huang et al. [5]. They proposed groundbreaking application of Transformer models in solar power prediction through their development of the Deep Learning Global Horizontal Irradiance Prediction (DL-GHIP) model, which combines Transformer layers with other components, such as convolutional layers and recurrent layers, to capture both spatial and temporal patterns in solar irradiance data.

The DL-GHIP model takes two types of inputs: satellite imagery, which provides spatial information about cloud cover and atmospheric conditions, and meteorological data, including variables like temperature, humidity, and wind speed that influence solar irradiance. The model's architecture leverages convolutional layers to process the satellite imagery and extract spatial features, recurrent layers (such as LSTMs) to model the temporal evolution of solar irradiance, and the core Transformer layers to capture long-range dependencies in the data through self-attention mechanisms. The outputs of these different components are then combined and passed through additional fully connected layers to produce the final global horizontal irradiance (GHI) prediction.

The authors evaluated the DL-GHIP model on several datasets, including ground-based radiation monitoring stations and satellite imagery, and found that it outperformed traditional statistical methods like ARIMA as well as other deep learning architectures like LSTMs and CNNs in terms of GHI prediction accuracy. This demonstrates the effectiveness of incorporating Transformer layers for modeling the complex spatial-temporal patterns inherent in solar irradiance data.

Transformer models for optimizing load dispatch in community microgrids

In their study, Wen et al. (2019)[35] investigated the application of Transformer models for optimizing load dispatch in community microgrids. Their approach integrates Transformer layers with static and dynamic covariates to enhance the accuracy of predicting solar power generation and load demand. At the heart of their innovative architecture are the Transformer layers, renowned for their proficiency in capturing intricate dependencies

within sequential data. Central to this capability is the self-attention mechanism embedded within Transformer layers, which allows the model to prioritize and weigh the relevance of different input features dynamically. This mechanism is pivotal in improving prediction accuracy by focusing on the most salient aspects of the data at any given time.

The architectural flow begins with the model ingesting various data inputs. These include time-series data pertaining to solar power generation and load demand, alongside static covariates such as building characteristics and geographic location, which remain constant over time. Dynamic covariates, such as weather conditions and energy consumption patterns, provide temporal context and fluctuate over time. The integration of both static and dynamic inputs is crucial as it enables the model to comprehensively understand and adapt to the multifaceted factors influencing the microgrid's operations. This dual integration ensures that the model not only comprehensively analyzes the current state of the microgrid but also anticipates future changes, thereby enhancing its forecasting capabilities.

Within the Transformer architecture, the input data undergoes an embedding process to convert categorical and numerical information into dense vector representations. These embeddings are then processed through multiple Transformer layers. Each layer consists of a self-attention mechanism followed by feed-forward neural networks. The self-attention mechanism allows the model to efficiently capture dependencies across different time steps, while the feed-forward networks facilitate learning complex, nonlinear relationships between input variables and desired outputs.

Overall, Wen et al.'s Transformer-based approach represents a significant advancement in microgrid optimization. The authors evaluated their model on real-world data from a community microgrid and found that it outperformed traditional optimization methods and other deep learning models, such as MLPs and LSTMs [35].

SL-Transformer

In their 2021 study Jian Zhu et al. [39], introduced the SL-Transformer, a specialized Transformer model tailored explicitly for the intricate demands of time-series power forecasting in wind and solar energy applications. The SL-Transformer was meticulously crafted to tackle the unique challenges posed by renewable energy data, including the irregularities and variability inherent in wind and solar power generation. Central to its design was the incorporation of domain-specific knowledge, a critical aspect highlighted by the authors. They integrated seasonal and diurnal patterns directly into the SL-Transformer's architecture, recognizing these factors as pivotal for accurate and reliable forecasting of renewable energy output.

The Transformer architecture operates by utilizing multiple layers to process sequential data. It consists of an encoder-decoder structure, where the encoder processes the input sequence and generates a contextualized representation, while the decoder generates the output sequence based on this representation. The Transformer model incorporates self-attention mechanisms in each layer, allowing it to focus on different parts of the input sequence when computing the representation of a specific token. This mechanism enables the model to capture long-range dependencies and complex patterns within the data. The inputs to the SL-Transformer model include historical energy production data, real-time weather forecasts, and domain-specific knowledge related to seasonal and diurnal patterns

in renewable energy generation. By integrating these inputs, the model can make accurate predictions of power generation, taking into account the dynamic and unpredictable nature of renewable energy sources. Through rigorous evaluations [39] demonstrated The SL-Transformer's ability to incorporate external factors and domain-specific information enhances its predictive capabilities and resilience against uncertainties.

Solar Transformer

The Solar Transformer [32] utilizes an architecture customized for forecasting solar irradiance. It comprises an encoder and decoder designed to process diverse data inputs effectively. The encoder integrates historical solar irradiance data, weather conditions, and positional information using self-attention mechanisms. These mechanisms identify key features across different domains, enhancing the model's ability to capture relevant patterns and relationships crucial for accurate predictions. Positional encoding ensures the model understands the sequential order and spatial context of the input data, crucial for distinguishing between data collected at various times and locations. Additionally, position-wise feed-forward networks refine data representations within the encoder, further improving prediction accuracy.

Alongside the encoder, the decoder simulates future solar irradiance levels based on the encoded representations. It combines latent space information from the encoder with additional inputs like day of the year information to predict solar irradiance for the following day accurately. The decoder incorporates normalization layers and residual connections to stabilize training and maintain data consistency, contributing to reliable prediction outcomes. The model's final output, determined by a linear activation function, provides straightforward forecasts of solar irradiance values.

Throughout its architecture, the solar Transformer employs optimization techniques aimed at enhancing prediction accuracy and model robustness. These include normalization layers and residual connections to ensure stable training and consistent data processing. Evaluation metrics such as mean absolute percentage error (MAPE) and mean squared error (MSE) validate the model's predictions against actual observations, ensuring its effectiveness in forecasting solar irradiance. Overall, the solar Transformer represents a sophisticated application of Transformer architecture tailored for renewable energy applications, leveraging advanced techniques to integrate diverse data sources and optimize predictive performance.

The findings highlighted the critical role of customizing Transformer architectures to suit the nuances of renewable energy forecasting. This study underscores the necessity of domain-specific adaptations to optimize model performance, providing valuable insights into tailoring advanced AI techniques to enhance the practical utility and effectiveness of forecasting tools within the renewable energy sector.

2.4 Hybrid deep learning models

Hybrid deep learning models combine various neural network architectures—such as CNNs, LSTMs, and Transformers—to leverage their unique strengths in data analysis. CNNs excel at extracting local features from spatial data, making them effective for modeling factors like cloud patterns affecting solar irradiance [5]. LSTMs are adept at capturing

long-range dependencies in sequential data, crucial for modeling the dynamic nature of solar irradiance and predicting short-term solar power output accurately. Transformers, with their ability to handle long-term dependencies, parallelize computations, and capture global patterns, further enhance the modeling capabilities of hybrid models [25]. This integration allows them to address data scarcity issues through transfer learning, enabling models trained in one location to generalize effectively to others with limited historical data.

In the study focusing on hybrid deep learning models for solar power forecasting [22], diverse combinations of CNN, LSTM, and TF models were extensively evaluated to capture the intricate patterns inherent in solar power data. The research utilized a dataset comprising four years of historical solar data from France [22], ensuring a robust analysis of real-world solar power generation dynamics. Among the evaluated models, the CNN–LSTM–TF hybrid model demonstrated superior performance, achieving a remarkable mean absolute error (MAE) of 0.551% when optimized with Nadam [22]. In contrast, the TF–LSTM model showed comparatively lower performance, highlighting the challenges in accurately predicting solar power outputs. Hybrid architectures such as CNN–TF, LSTM–TF–CNN, TF–CNN–LSTM, and TF–LSTM–CNN effectively captured both temporal variations and local–global interactions within the solar data, outperforming traditional methods like persistence, backpropagation neural networks (BPNN), and radial basis function neural networks (RBFNN). The study underscores the significance of model and optimizer selection, emphasizing the impact of choices such as Adam, Nadam, Adamax, and RMSprop on forecast accuracy. By leveraging the strengths of LSTM for handling long-term dependencies, CNN for extracting spatial features, and Transformers for managing global relationships through attention mechanisms [5, 25], these hybrid models not only enhance predictive accuracy but also hold potential for advancing the integration of solar energy into modern electrical power systems.

2.5 Comparative analysis of solar power forecasting methods

Accurate forecasting of solar power generation is essential for effectively integrating renewable energy into the grid. To enhance the precision and reliability of these forecasts, several deep learning models have been developed. The following table offers a comparative analysis of various deep learning models used in time series solar power forecasting.

Table 2.1: Compilation of deep learning models for time series solar power forecasting

Model	Description	Key Findings
Transformer (TF)	Employs self-attention mechanism for capturing global dependencies in time series data.	Superior for long sequences but might have limitations in capturing sequential patterns compared to RNNs.
TFT (Transformer-based forecasting)	Demonstrates superior performance compared to LSTMs, CNNs, and traditional methods.	Effective for long-term forecasts; however, they are computationally demanding.
LSTM-Transformer	Combines strengths of LSTMs (capturing long-term dependencies) and Transformers (global patterns).	Leverages benefits of both architectures but requires complex integration.
CNN (Convolutional Neural Network)	Efficient at identifying local patterns and critical parts of time series data.	Effective for short-term forecasting of solar power but struggle with long-term dependencies.
LSTM (Long Short-Term Memory)	Designed for capturing long-term dependencies with memory cells.	Effective for long-term forecasting but computationally expensive.
Hybrid Models	Combine strengths of different architectures for broader pattern recognition.	Achieve high accuracy through diverse pattern recognition but require complex implementation.
CNN-LSTM	Integrates CNNs and LSTMs for ultra-short term solar power forecasting.	Effective for real-time forecasting but computationally intensive.
GRU (Gated Recurrent Unit)	Focuses on long-term solar radiation forecasting.	Suitable for long-term forecasting but less popular in solar power forecasting compared to LSTMs and Transformers.
Satellite-Based Models	Utilize satellite data for short-term solar forecasting.	Effective for short-term forecasting but depend on good quality satellite data.
NWP (Numerical Weather Prediction) Models	Focus on medium-range and long-range solar forecasting using weather data.	Effective for medium to long-range forecasting but pose integration challenges.
DL-GHIP (Deep Learning Global Horizontal Irradiance Prediction)	Pioneering application of Transformers in solar power prediction.	Initial exploration of Transformers demonstrates promising results for solar power forecasting.
SL-Transformer	Tailored Transformer model for wind and solar power forecasting.	Integrating domain knowledge enhances Transformer effectiveness in renewable energy forecasting.

2.6 Discussion

The integration of anomaly detection into forecasting frameworks using Transformer models marks a significant advancement in time series analysis. Traditionally treated as distinct tasks with separate methodologies, the fusion of these capabilities within a unified Transformer-based framework offers compelling advantages. Transformers are renowned for their ability to capture complex temporal relationships through self-attention mechanisms, which makes them adept at both forecasting future trends and pinpointing anomalies within data streams. By embedding anomaly detection directly into the forecasting process, these models can leverage shared representations and learn anomaly-specific features from the data itself, thereby enhancing overall accuracy and reliability.

Looking ahead, the evolution of Transformer models for integrated anomaly detection and forecasting could focus on several key areas. Future developments may include refining models to dynamically adapt to changing data distributions and optimizing multi-task learning objectives to maintain a balance between prediction accuracy and anomaly detection sensitivity. Enhancing model interpretability through transparent decision-making processes will be crucial for fostering trust and usability in practical applications. Additionally, addressing scalability challenges and exploring hybrid approaches that combine Transformers with complementary techniques like GANs or VAEs could further extend the capabilities of these integrated frameworks. By advancing in these dimensions, Transformer models not only promise improved predictive performance but also bolster resilience

in real-time anomaly detection, making them invaluable across diverse domains requiring robust time series analysis capabilities.

2.7 Conclusion

The rapid growth of solar energy as a renewable alternative to fossil fuels necessitates accurate solar power forecasting for optimal grid integration and utilization. Transformer models, with their ability to capture long-range dependencies and handle high-dimensional data, offer significant advantages over traditional statistical methods and other deep learning architectures. This chapter has reviewed the role of AI in solar forecasting, highlighted the effectiveness of Transformer models, and discussed their application in solar power prediction. Future research should focus on refining Transformer architectures, developing domain-specific models, and leveraging transfer learning to enhance the accuracy and reliability of solar power forecasts.

Chapter 3 Methodology

3.1 Introduction

The aim of this chapter is to present our proposed Transformer-based model for forecasting solar power generation. An outline of our methodology is illustrated in Figure 3.1. We first give an overview of the solar power generation process, which is the specific context of our work, providing the input to our model. Then, we describe the data we used for validating our model, highlighting the data preparation techniques. Additionally, The tools used in the implantation are outlined. By the end, a detailed description of the proposed network architecture is provided.

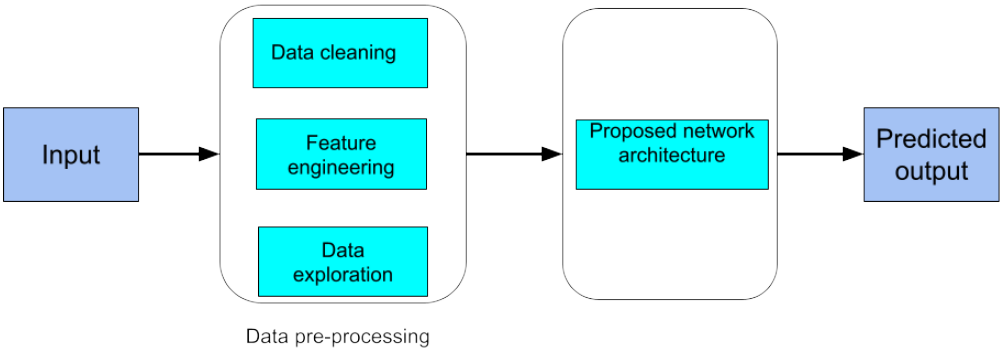


Figure 3.1: Main blocks of our approach.

3.2 Solar power generation

Solar power generation and accurate prediction have far-reaching impacts on the energy industry and the environment. By optimizing maintenance schedules, minimizing downtime, and extending the lifespan of components, solar power systems can operate more efficiently, leading to significant cost savings and enhanced grid stability.

Solar power plants are comprised of various key components, as illustrated in Figure 3.2, including PV modules, Grid, Inverters, Transformer, and many others.

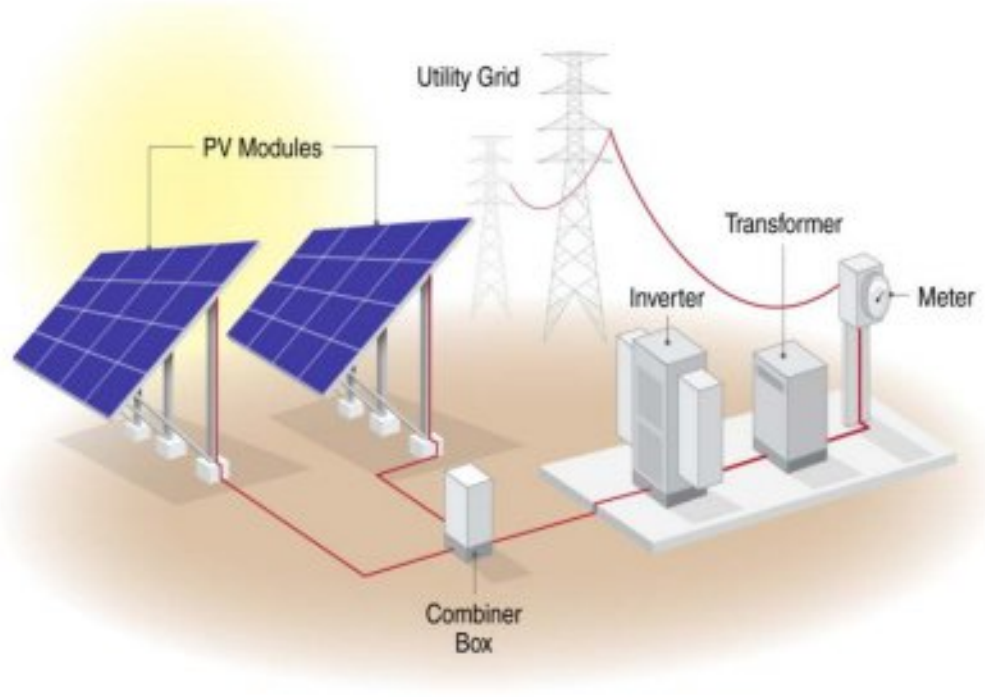


Figure 3.2: Solar power plant overview [4].

Solar power generation involves a series of steps to convert sunlight into usable electricity through photovoltaic systems, The process as shown in Figure 3.3 begins with pv arrays, which Utilize sunlight to generate direct current (DC) electricity. This initial stage is crucial as it directly impacts the amount of electricity generated based on the intensity of sunlight and the efficiency of the PV modules.

Once generated, the DC electricity from multiple PV modules is typically routed to a combiner box. This component serves as a central point where the outputs of several PV strings are combined into a single DC output. It also facilitates easy disconnection of the DC power for maintenance or safety purposes, ensuring operational efficiency and safety compliance.

After leaving the combiner box, the DC electricity undergoes inversion. Inverters are fundamental units that alter DC electricity into alternating current (AC) electricity. This transformation is necessary because AC electricity is the standard form used in most homes and businesses for powering appliances, lighting, and other electrical devices.

To facilitate efficient distribution and transmission, the voltage of the electricity may need to be adjusted. Transformers are used to modify voltage levels as necessary, stepping them up or down. This ensures that electricity can be transmitted over long distances with minimal loss and distributed at appropriate voltage levels for consumer use.

In some solar power systems, a battery bank is integrated to store excess electricity generated during periods of high sunlight. The battery bank plays a critical role in storing DC electricity for later use, providing energy resilience and flexibility to the system

Throughout this process, the electricity meter plays a critical role in measuring the

amount of electricity generated by the solar power system. It tracks both the electricity produced by the PV array and the electricity consumed by the premises. This data is essential for monitoring the system's performance, ensuring accurate billing, and enabling net metering arrangements where excess electricity can be fed back into the utility grid.

The interaction with the utility grid is significant in solar power generation. Excess electricity generated beyond immediate consumption needs can be fed back into the grid, allowing for net metering. This practice not only offsets electricity costs for consumers but also contributes to grid stability and reduces overall reliance on non-renewable energy sources.

To maintain optimal performance and reliability, solar power systems are equipped with monitoring systems that utilize sensors to continuously track parameters such as solar irradiance, temperature, and power output. These sensors provide real-time data that helps operators and maintenance personnel optimize system efficiency, identify potential issues promptly, and ensure consistent electricity production.

In conclusion, the cycle of solar power generation—from sunlight capture to electricity distribution and grid interaction—is carefully orchestrated to maximize efficiency, reliability, and environmental benefits. By harnessing solar energy, these systems play a crucial role in sustainable energy production, reducing greenhouse gas emissions, and promoting a cleaner energy future.

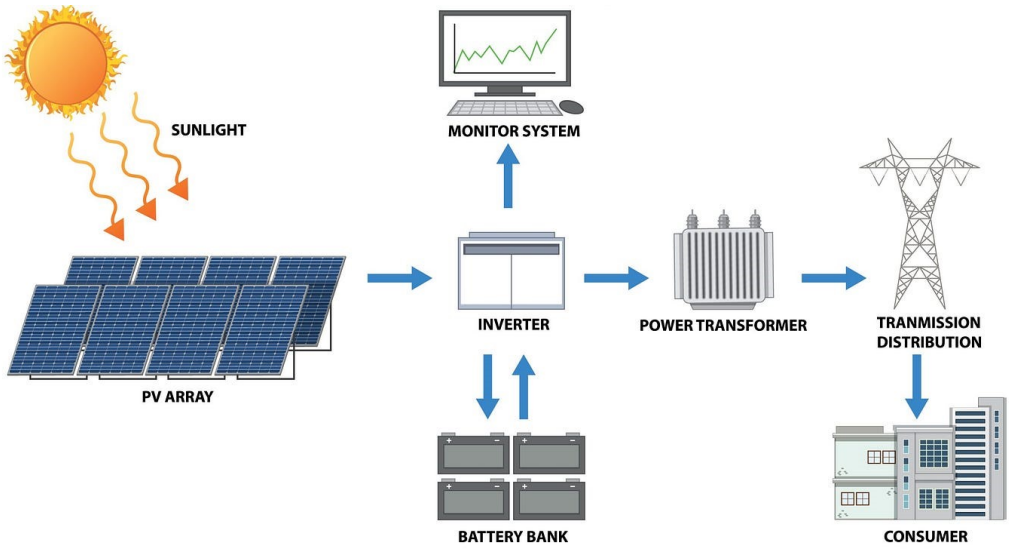


Figure 3.3: Solar power generation process [20].

3.3 Data description and preparation

The dataset used in this study is sourced from Kaggle [10]. It offers an extensive analysis of the performance of two solar power facilities in India, namely Plant 1 near Gandikotta, Andhra Pradesh, and Plant 2 near Nasik, Maharashtra. Spanning a duration of 34 days, this dataset contains information about solar power generation. The data was collected in the year 2020, with recordings taken at 15-minute intervals, resulting in over 60,000 timestamps for each plant.

The dataset consists of two primary types of files for each plant: power generation data and weather sensor readings. The power generation data is captured at the individual inverter level, where each inverter is connected to multiple lines of solar panels to get the following records:

- DC_POWER: The amount of DC power produced by the specified inverter (SOURCE_KEY) within a 15-minute interval, measured in kilowatts (kW).
- AC_POWER: Represents the amount of AC power generated by the specified inverter (SOURCE_KEY) during a 15-minute interval, measured in kW.
- DAILY_YIELD: Shows the cumulative sum of power generated by the inverter on that particular day up to the recorded time.
- TOTAL_YIELD: Reflects the total power yield generated by the inverter up to the recorded time, since its installation.

On the other hand, the weather sensor data pertains to the entire plant, utilizing a single array of strategically positioned sensors situated across the facility to get the following records:

- Ambient temperature: Represents the temperature of the plant's environment, measured at the sensor's location.
- Module temperature: Indicates the temperature reading of a specific module (solar panel) attached to the sensor panel.
- Irradiation: Reflects the amount of irradiation received at the plant's location during each 15-minute interval.

Usually, raw data acquired from fieldwork often contains inconsistencies. Therefore, data preprocessing is a crucial stage aimed at enhancing data consistency, accuracy, and usability. This process plays a vital role in improving the performance and reliability of models and analytical methods. In our case, the key steps involved in data preprocessing include:

3.3.1 Data cleaning

Data cleaning is crucial for ensuring the quality of data used in model training. The process aims to identify and rectify errors, inconsistencies, or inaccuracies present in the dataset. In our case, we noted that sunlight consistently occurs from 5 AM to 6 PM, with only a few hours lacking sunlight. Consequently, values outside this period were set to 0, except for temperature data which remains unaffected.

To handle missing daily yields after sunset, we replaced them with the most recent non-null value from the same day, taking into account the cumulative nature of the data. For other missing values, we employed linear interpolation. This method utilizes the sequential nature of time-series data, assuming that observations follow linear trends throughout the day. Although this approach may not perfectly capture peak values at noon, the small number of missing entries ensures minimal impact on the overall dataset integrity. Our primary aim was to fill missing values as accurately as possible based on expected patterns

in the data, and used the following methods for filling missing values:

1. **Interpolation:** is a technique to approximate unknown values that are positioned between already known data points. In the context of data preprocessing, it is specifically beneficial in filling in any missing values in a dataset, particularly when dealing with time series data, in which retaining the temporal sequence is crucial. For instance, when encountering missing values within a time series dataset, linear interpolation can be employed to estimate these values. This is achieved by connecting straight lines between the closest known values and determining the intermediate points along this line.

2. **Imputation:** is the process of replacing missing data with substituted values. The objective is to provide a complete dataset without losing significant information or introducing bias. The two used methods in our preprocesssing are:

   Zero imputation: This approach is applicable when it is reasonable to interpret the absence of data as indicating a zero value, for example, null values outside of sunlight hours are set to zero.

   Forward fill: This method involves replacing missing values with the latest available non-missing value that precedes it. For example, for missing daily yields after sunset, forward fill technique is employed.

Both techniques serve to handle missing data, with interpolation focusing on estimating within a sequence of data points and imputation ensuring dataset completeness for analysis. By applying these techniques in combination, the dataset is completed to effectively represents the underlying patterns with accuracy. Finally, we conducted a final verification to guarantee the absence of any further missing data.

3.3.2 Normalization

Normalization is employed to standardize numerical feature values to a uniform scale, ensuring that variations in value ranges are preserved while enhancing model performance. In our case, Min-Max scaling was used to normalize the data.

Min-Max Scaling is a data transformation technique that rescales the data to fit within a range of 0 to 1 (or sometimes -1 to 1). The formula used for Min-Max Scaling is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

where:

$$\begin{aligned} x &: \text{Original value} \\ \min(x) &: \text{Minimum value of the feature} \\ \max(x) &: \text{Maximum value of the feature} \\ x' &: \text{Normalized value} \end{aligned}$$

3.3.3 Feature engineering

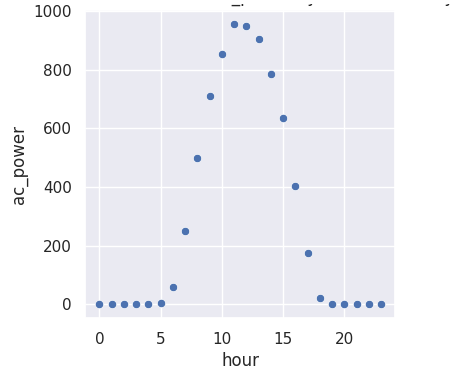
To enhance the predictive capabilities of our model, we introduced supplementary features to capture DC power, AC power, and yield at different levels. These additional features allowed us to calculate values at specific time intervals, such as the 15-minute, hourly, daily, and inverter levels. By incorporating these time-based features into the dataset, we not only increased the granularity of the data but also enhanced its interpretability. This facilitated a more comprehensive analysis of temporal patterns and trends, enabling us to explore relationships between solar generation, weather data, and their fluctuations over time.

3.3.4 Data exploration

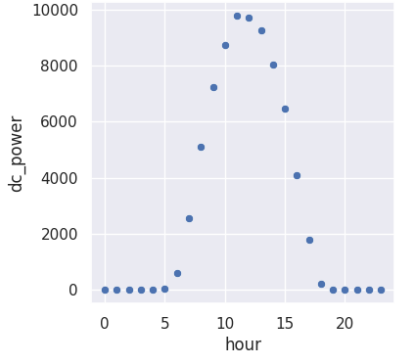
Visualizing and inspecting data is crucial for gaining deeper insights into the problem at hand. Data exploration involves reviewing the distributions of various features.

Below are graphical representations that illustrate the distribution of selected features for both plants:

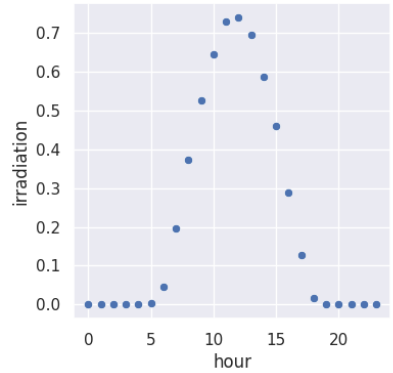
- **Features distribution for plant 1**:



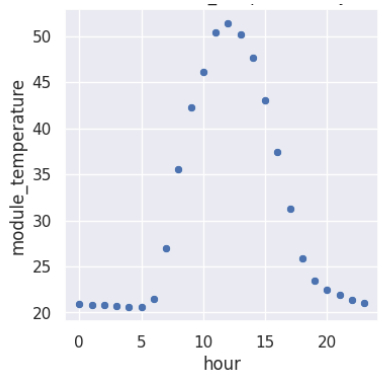
(a) AC power by hour of the day.



(b) DC power by hour of the day.



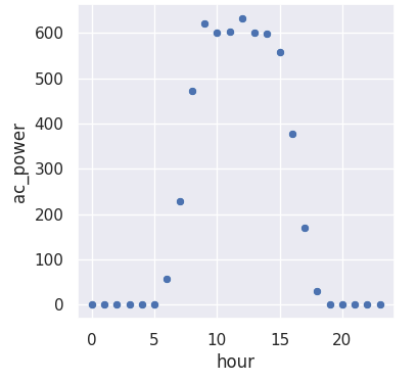
(c) Irradiation by hour of the day.



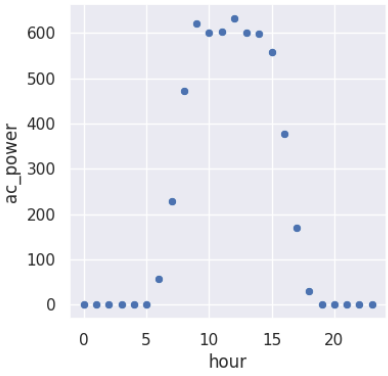
(d) Module temperature by hour of the day.

Figure 3.4: Features distributions of Plant 1.

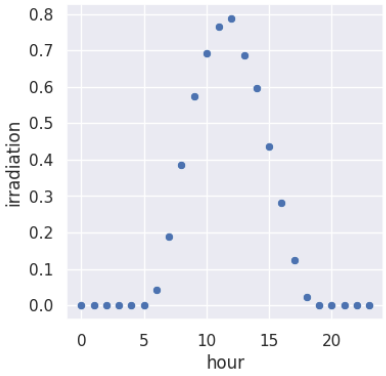
- **Features distribution for Plant 2**:



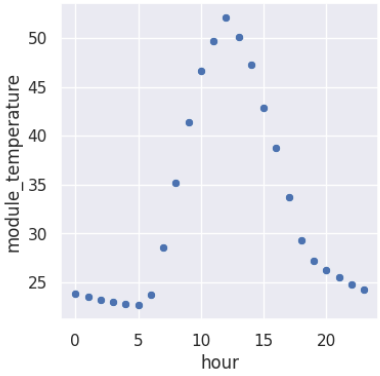
(a) AC power by hour of the day.



(b) DC power by hour of the day.



(c) Irradiation by hour of the day.



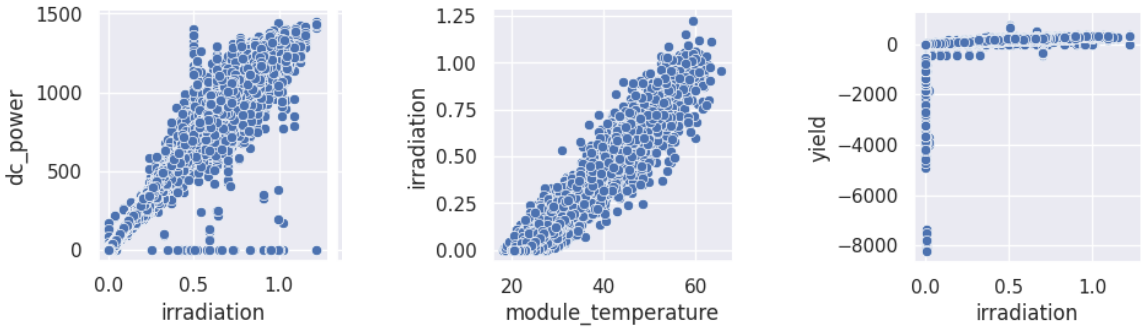
(d) Module temperature by hour of the day

Figure 3.5: Features distributions of Plant 2.

Upon reviewing the distributions, a few key observations were made. Firstly, it was evident that many features displayed skewed distributions, indicating the need for transformation to achieve more normal distributions before modeling. However, it was noted that the abundance of zero values among features significantly contributed to the skewed distributions. Secondly, the recently introduced yield and hourly_yield features showed extreme outlier values. A notable finding specific to Plant 2 was the occurrence of days with zero power generation. This variability underscores the importance of understanding and potentially addressing periods of inactivity in the data. Lastly, the dc_ac_ratio was either 0 or 1, meaning it cannot be used for forecasting. This also implies that either dc_power or ac_power can be used, while disregarding the other feature. The same rule applies to all features derived from the relationship between dc_power and ac_power. Consequently, it was decided to keep the dc_power features, discard the ac_power features, and exclude all dc/ac ratio features.

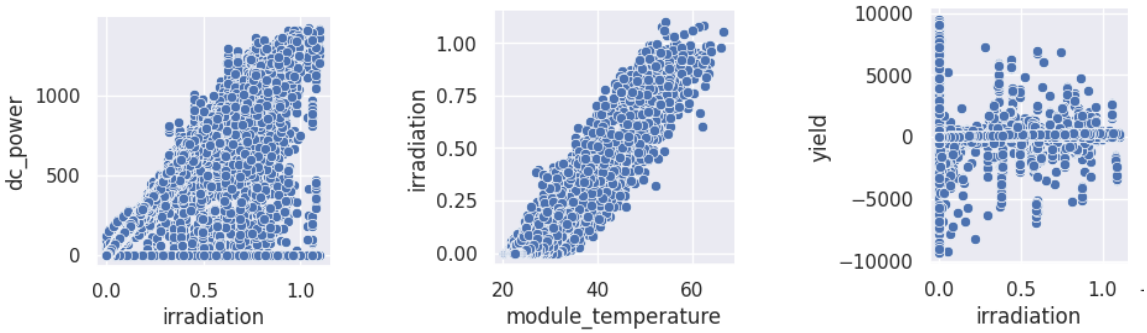
Regarding the presence of zero values in power generation as it can be seen in the distribution of ac_power and dc_power for both plants, it is important to note that these values often occur during hours with no sunlight or on days where no power was generated. These zeros present challenges when it comes to scaling or standardizing the data for modeling purposes. To solve this issue, We have decided to remove the rows corresponding to periods without sunlight. After all, forecasting during these times becomes irrelevant since no power is produced. By removing these rows, a significant portion of the zero values can be eliminated, thereby reducing or eliminating the skewness in the distributions.

Subsequently, we proceeded to investigate the bivariate relationships between various features for both plants. This exploration led us to generate the following graphical representations:



(a) (b) (c)

Figure 3.6: The bivariate relationships between features for Plant 1.



(a) (b) (c)

Figure 3.7: The bivariate relationships between features for Plant 2.

As it can be seen from the plots, temperature and irradiation are strongly correlated, which is not surprising. However, the distribution of yield appeared peculiar, making it challenging to identify clear trends. On the other hand, it seemed that DC power could serve as a more reliable indicator of output. It is plausible that there are other factors influencing yield that are not captured in the dataset, potentially leading to adjustments in yield throughout the day due to unobserved phenomena. The variability in Plant 2 data overshadowed the relationship between power output and temperature/irradiation. After thorough exploration of the data, the decision was made to utilize DC power as the target variable for forecasting instead of yield, and irradiation instead of temperature.

3.3.5 Combining the two datasets

Certain relationships between features, such as the correlation between irradiation and yield, are likely to be consistent across both plants. In such cases, it would be logical to merge the data from both plants to increase the sample size and improve the learning capabilities of the model. On the other hand, there are unique relationships that exist specifically between inverters.

We combined both datasets by aligning their date-times through a left join. In this process, all rows from the left DataFrame (solar generation data) were kept in the merged DataFrame (plant1 or plant2), and any matching rows from the right DataFrame (weather data) were added if available. In cases where no match was found in the right DataFrame, corresponding columns were filled with NaN (Not a Number) values. In this case, the merge was performed based on the 'date_time' column, which represents the timestamp of the data readings.

3.4 Proposed model

In our project, we implemented a Transformer-based model for predicting solar panel power outputs over time, taking cues from the typical structure of the Transformer model. This model aims to forecast the DC power output for the upcoming time interval using past data. Our model architecture is detailed in the following.

3.4.1 Network architecture

In this section, we will delve into the modifications we did to the original Transformer model to make it suitable for time series forecasting and anomaly detection by presenting our final architecture.

The model's architecture is organized in a specific sequence, starting with processing input data, then using Transformer encoder blocks to extract temporal features, followed by pooling and feature extraction, and finally, a Kolmogorov Arnold Networks (KAN) layer for additional feature transformation and output generation. This structured order allows the model to effectively identify temporal patterns and generate accurate predictions from the input time series data, as illustrated in Figure 3.8.

3.4.1.1 Input layer

The input layer of the model accepts sequences of historical time steps, each containing several features. This well-organized input enables the model to identify patterns and correlations over time. The input shape is represented as (T, D), where T represents the number of past time steps considered for the prediction, and D indicates the number of features.

3.4.1.2 Positional encoding

In the context of Transformers for time series analysis, it is crucial to incorporate positional information since the order of the data matters. This is typically achieved by encoding the positions of the input time series as vectors and then combining these vectors with the input time series as additional inputs to the model. This ensures that the model is aware of the temporal relationships within the data, allowing it to effectively capture patterns and trends in the time series. In our case, static positional encoding is incorporated into the input sequences. We calculated the positional encoding as follow:

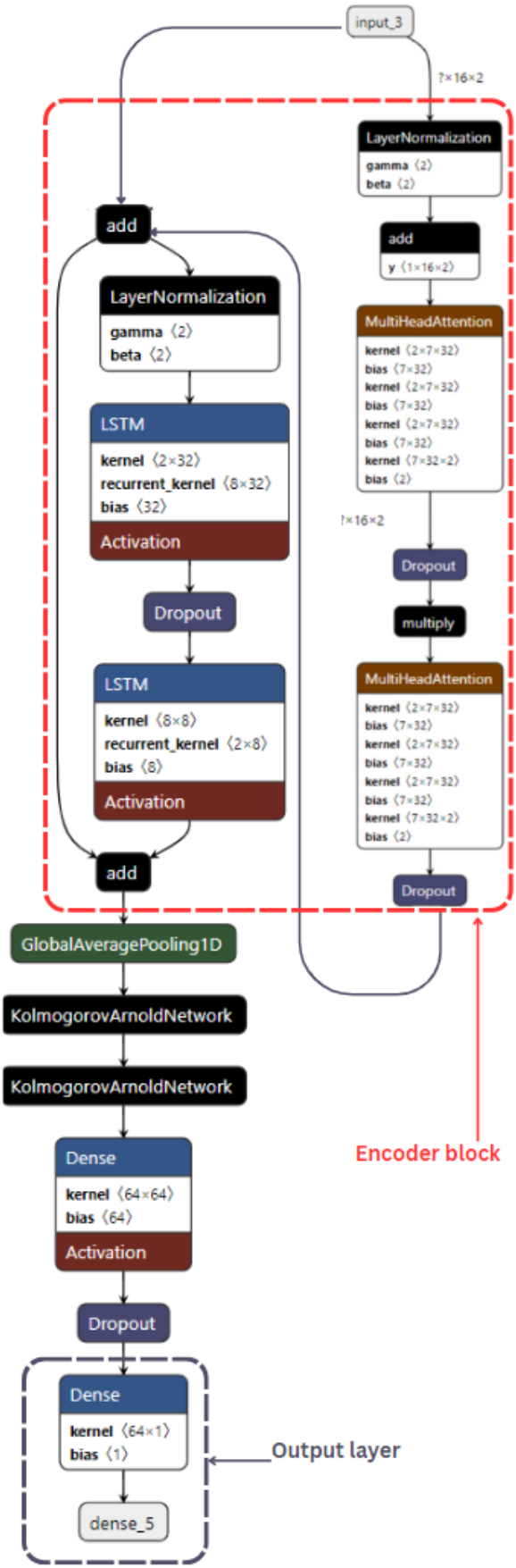


Figure 3.8: Network architecture of the proposed model.

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 16, 2)]	0	[]
layer_normalization_4 (LayerNormalization)	(None, 16, 2)	4	['input_5[0][0]']
tf.__operators__.add_6 (TFOpLambda)	(None, 16, 2)	0	['layer_normalization_4[0][0]']
multi_head_attention_4 (MultiHeadAttention)	(None, 16, 2)	2466	['tf.__operators__.add_6[0][0]', 'tf.__operators__.add_6[0][0]']
dropout_12 (Dropout)	(None, 16, 2)	0	['multi_head_attention_4[0][0]']
tf.math.multiply_2 (TFOpLambda)	(None, 16, 2)	0	['dropout_12[0][0]']
multi_head_attention_5 (MultiHeadAttention)	(None, 16, 2)	2466	['tf.math.multiply_2[0][0]', 'tf.math.multiply_2[0][0]']
dropout_13 (Dropout)	(None, 16, 2)	0	['multi_head_attention_5[0][0]']
tf.__operators__.add_7 (TFOpLambda)	(None, 16, 2)	0	['dropout_13[0][0]', 'input_5[0][0]']
layer_normalization_5 (LayerNormalization)	(None, 16, 2)	4	['tf.__operators__.add_7[0][0]']
lstm_5 (LSTM)	(None, 16, 8)	352	['layer_normalization_5[0][0]']
dropout_14 (Dropout)	(None, 16, 8)	0	['lstm_5[0][0]']
lstm_6 (LSTM)	(None, 16, 2)	88	['dropout_14[0][0]']
tf.__operators__.add_8 (TFOpLambda)	(None, 16, 2)	0	['lstm_6[0][0]', 'tf.__operators__.add_7[0][0]']
global_average_pooling1d_2	(None, 16)	0	['tf.__operators__.add_8[0][0]
kolmogorov_arnold_network_4 (KolmogorovArnoldNetwork)	(None, 64)	1024	['global_average_pooling1d_2[0][0]']
kolmogorov_arnold_network_5 (KolmogorovArnoldNetwork)	(None, 64)	4096	['kolmogorov_arnold_network_4[0][0]']
dense_6 (Dense)	(None, 64)	4160	['kolmogorov_arnold_network_5[0][0]']
dropout_17 (Dropout)	(None, 64)	0	['dense_6[0][0]']
dense_7 (Dense)	(None, 1)	65	['dropout_17[0][0]']

Total params: 14725 (57.52 KB)
Trainable params: 14725 (57.52 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 3.9: Summary of the proposed model.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{3.2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{3.3}$$

The input sequences are subsequently directed into the core of the model, the Transformer encoder block.

3.4.1.3 Encoder block

Our proposed model is comprised of a Transformer encoder-only structure augmented with LSTM layers. In addition to the elements outlined in Part 2, we have also included:

- **LSTM layers**: The attention mechanism within Transformers effectively detects and assigns significance to pivotal associations within the sequence. Integrating this with LSTMs, which excel at preserving sequence context, allows the model to harness the advantages of both architectures, This ensures comprehensive coverage of temporal patterns by the LSTM, enhancing the model's ability to capture nuances across the dataset, in synergy with the attention mechanism.

- **Dropout**: Dropout is a regularization technique utilized in neural networks to mitigate overfitting and enhance the model's generalization capabilities. In each training iteration, dropout randomly eliminates a certain portion of nodes, setting their values to zero, which compels the remaining neurons to learn more robust features and thereby prevent overfitting.

- **Residual connections**: These connections are crucial to prevent the problem of vanishing gradients, which can arise in deep neural networks during backpropagation. By integrating residual connections, the Transformer architecture guarantees the preservation of the gradient signal, particularly when utilizing activation functions such as ReLU, which can yield zero gradients in specific scenarios. Furthermore, the inclusion of residual connections sustains local information flow within the Transformer layer stack, ensuring that the input tokens representations retain their original context as the data traverses multiple layers

3.4.1.4 Global Average Pooling

The output generated by the Transformer blocks consists of a series of hidden states that encode information regarding the input time series, which is then enhanced by positional encoding. Global Average Pooling (GAP) is applied to these hidden states to condense the sequence dimensionality into a single vector by averaging the features across all time steps. This pooling operation effectively captures the overall features of the input sequence, compressing the learned information into a concise beneficial form.

3.4.1.5 Kolmogorov-Arnold Network

The integration of the Kolmogorov-Arnold Network (KAN) layer in our model marks a paradigm shift, transforming feature learning within the Transformer architecture.

Inspired by the Kolmogorov-Arnold representation theorem, the authors of the groundbreaking KANs paper [13], released in May 2024, KANs as a revolutionary class of Neural Network building blocks. KANs represent a significant advancement over MLPs, offering heightened expressiveness, reduced susceptibility to overfitting, and improved interpretability.

Unlike MLPs, which rely on fixed activation functions at nodes and often suffer from overfitting due to their high parameter count and inflexible architecture, KANs use adaptive B-spline activation functions. Each edge weight in KANs is replaced by a learnable, single-variable B-spline function, as depicted in Figure 3.10. This innovative approach enables KANs to capture complex relationships between inputs and outputs more effectively, leading to enhanced performance and expressiveness.

- **KAN formula**:

Splines solve the problem of interpolating smoothly between multiple points. Splines tackle this issue by fitting piecewise polynomial functions to sections between data points. KANs use B-splines, which are piecewise polynomials that offer local control and smooth interpolation between data points, addressing issues with traditional activation functions.

For any continuous function: $f : \mathbb{R}^n \to \mathbb{R}$, there exist continuous functions ϕ_{qk} and ψ_k such that:

$$f(x_1, x_2, \ldots, x_n) = \sum_{k=0}^{2n+1} \psi_k \left(\sum_{q=1}^{n} \phi_{qk}(x_q) \right) \tag{3.4}$$

where:

- $x = (x_1, x_2, \ldots, x_n)$ is the input vector in $\mathbb{R}^n$,
- ϕ_{qk} are continuous functions of a single variable, transforming each input.
- ψ_k are continuous outer functions applied to the inner transformations.

3.4.1.6 Output Dense Layer

The output from the KAN layers is passed through a final dense layer, also known as fully fulled connected layer. It converts the condensed features into the anticipated DC power output for the subsequent time step, delivering the predicted value that the model was trained to predict.

The output dimension of the final dense layer corresponds to the number of time steps ahead the model aims to forecast, in our case is 16 time steps (considering 15-minute intervals).

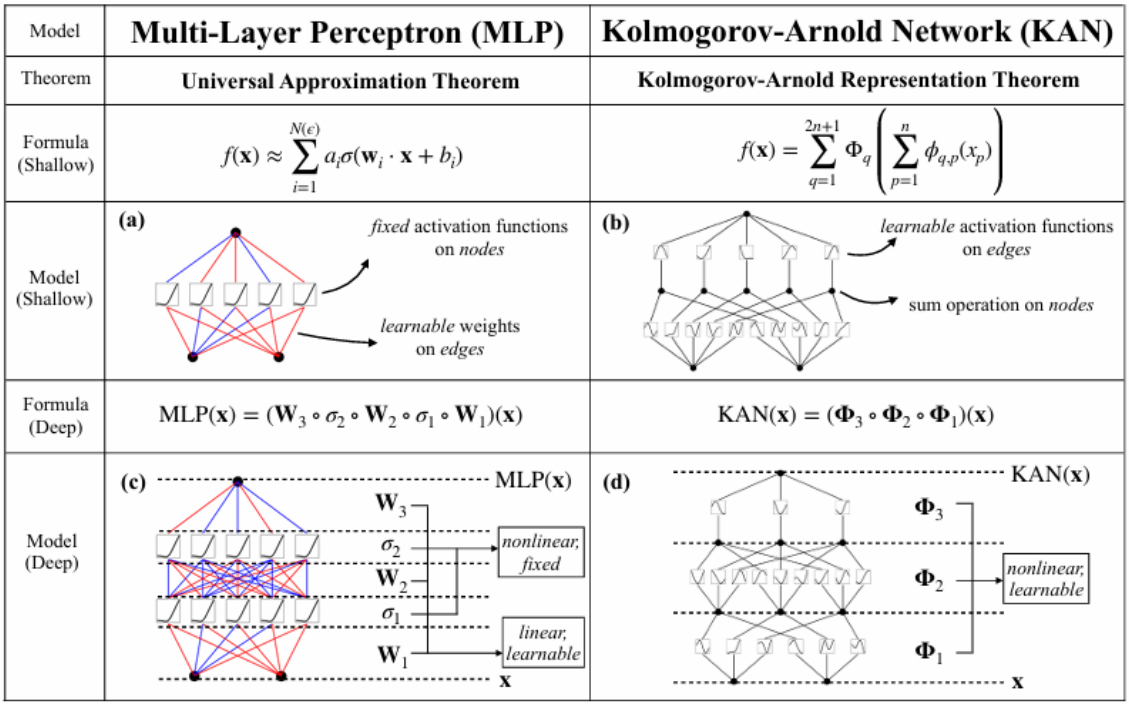


Figure 3.10: MLPs vs KANs [13].

3.4.2 Training procedure

The model training process involves compiling the model using the MSE loss function and the Adam optimizer, which employs a learning rate of 0.05. MSE is particularly suitable for regression tasks as it penalizes larger errors, encouraging precise predictions. Meanwhile, the Adam optimizer's adaptive learning rate characteristics aid in effective training. The model is trained over 200 epochs with a batch size of 32, which strikes a balance between training stability and computational efficiency. To ensure a realistic evaluation of performance on unseen data, the dataset is divided such that the first half is used for training and the second half for validation which contains data that the model hasn't seen before. The training progress is tracked and displayed with verbose output, and the duration of the training process is recorded to provide insights into computational efficiency.

3.5 Implementation tools

3.5.1 Pyhton

Python was selected as the primary programming language for the software environment due to its user-friendly nature, simplicity, readability, and its widespread usage in the field of artificial intelligence and it boasts a wide range of libraries dedicated to deep learning that offer high-level abstractions that simplify the process of designing and training neural networks:

1. **Keras:** is an open-source library that utilizes deep learning frameworks like TensorFlow for efficient processing. It leverages their performance optimizations and hardware support, including GPU accelerations, to provide faster processing times. The high-level Application Programming Interface (API) offered by Keras is intended for

developing and training deep neural networks, allowing developers and researchers to experiment effectively with different models and architectures.

2. **NumPy:** is an essential package used for scientific computing in Python. It offers extensive support for handling large arrays and matrices with multiple dimensions. Additionally, NumPy provides a vast range of high-level mathematical functions specifically designed to perform operations on these arrays efficiently.

3. **Matplotlib:** is a Python library used for plotting, which is an extension of the numerical mathematics library NumPy. The pyplot module within Matplotlib offers a convenient interface similar to MATLAB, allowing users to generate static, animated, and interactive visualizations in Python.

4. **Pandas:** is a robust data manipulation and analysis library used in Python. It includes data structures such as Series (primarily one-dimensional) and DataFrame (two-dimensional) that are highly beneficial for efficient management and analysis of data. Pandas comes equipped with several tools for tasks such as data cleaning, aggregation, visualization, and manipulation.

5. **Scikit-learn:** is a machine learning library available as an open-source package for Python. It enables to mine and analyze data effectively through its user-friendly and efficient tools. It is built on top of NumPy, SciPy, and matplotlib, and designed to interact seamlessly with Python's numerical and scientific libraries by providing a consistent interface for various machine learning algorithms.

3.5.2 Kaggle

It is an online community that brings together data scientists and machine learning engineers. It provides a platform for users to discover, publish, and access datasets for building AI models. Moreover, they can connect, collaborate, and compete on data-driven challenges.

3.5.3 Google collaboratory

Colaboratory ("Colab" for short) simplifies the process of setting up and installing complex configurations by operating directly within the browser. It comes with pre-installed Python libraries and frameworks that can be used without any additional setup requirements. It is designed based on the Project Jupyter code and can host Jupyter notebooks without the need for any software installations on your device.

3.6 Conclusion

In this chapter, we thoroughly explained the methodology used to develop our Transformer-based model for time series forecasting. We provided an in-depth review of the proposed network architecture, highlighting its various components and how it contrasts with the traditional Transformer model. This sets the stage for the performance evaluation of the model in Chapter 4.

Chapter 4 Results and discussion

4.1 Introduction

In Chapter 4, we delve into the experiments, results, and discussions regarding the optimization of our Transformer model and conduct a comparative analysis. The chapter encompasses hyperparameter tuning, architectural improvements, and the rationale behind the configuration of the final model. It offers a detailed examination of the thorough process involved in refining the model to attain precise and effective forecasting and anomaly detection.

4.2 Evaluation metrics

Evaluation metrics are critical in measuring the performance of a machine learning model. There are various metrics available, and the choice of a particular metric depends on the specific task. For our task the primary metrics include MAE, MSE and RMSE for the forecasting task and confusion matrix and confusion metrics for fault prediction.

1. **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the actual values and the predicted values. It provides a linear score which means all the individual differences are weighted equally in the average, lower MAE values signify better predictive performance. It is calculated as follow:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.1}$$

   where:

   - n is the number of observations,
   - y_i is the actual value,
   - $\hat{y}_i$ is the predicted value.

2. **Mean Squared Error (MSE):** MSE quantifies the average of the squared discrepancies between predicted and actual values, imposing significant penalties for larger deviations. It is beneficial when larger errors are particularly undesirable and should be penalized more heavily.

The formula for MSE is:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

where:

- n is the number of data points.
- y_i is the true value of the target variable for the i-th data point.
- $\hat{y}_i$ is the predicted value of the target variable for the i-th data point.

3. **Root Mean Squared Error (RMSE):** RMSE is calculated by taking the square root of MSE. This adjustment allows the error to be expressed in the same units as the output variable, enhancing interpretability. It is frequently used when it is important to directly compare the scale of prediction errors with the scale of the actual data to be directly compared to the scale of the data. The formula for the RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.3}$$

where:

- n is the number of observations,
- y_i is the actual value,
- $\hat{y}_i$ is the predicted value.

4. **Confusion Matrix:** A confusion matrix is a tabular representation used to assess the effectiveness of a machine learning model. These performance metrics are crucial when the model is tasked with accurately detecting specific patterns, such as anomalies, in our case. The confusion matrix consists of four possible outcomes for each class:

   - **True Positive (TP):** Instances that are correctly identified as positive by the model.
   - **True Negative (TN):** Instances that are correctly identified as negative by the model.
   - **False Positive (FP):** Instances that are incorrectly identified as positive by the model.
   - **False Negative (FN):** Instances that are incorrectly identified as negative by the model.

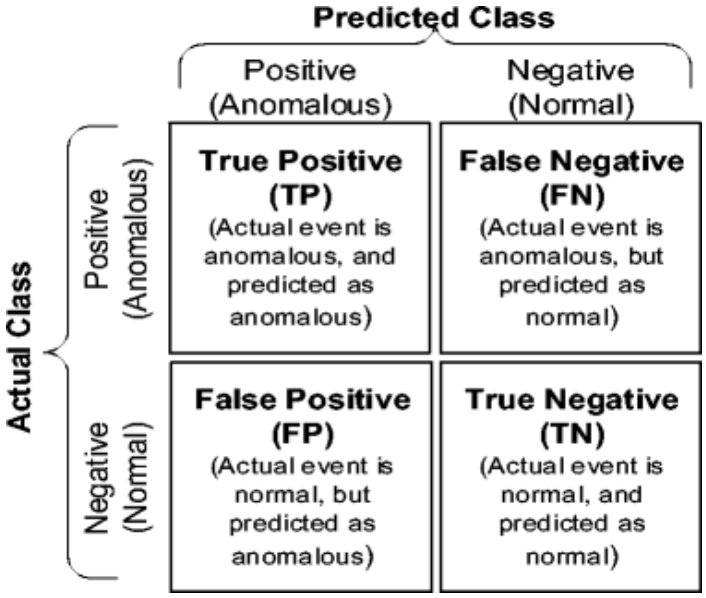


Figure 4.1: Confusion matrix for anomaly detection.

These four outcomes form the basis for calculating various performance metrics, such as precision, recall, F1 score, false positive rate, and false negative rate. These metrics provide a comprehensive understanding of the model's ability to accurately classify instances and minimize false detection, making them particularly valuable in anomaly detection scenarios.

5. **Precision:** Precision assesses the proportion of correctly classified positive instances out of all predicted positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.4}$$

6. **Recall:** Recall evaluates the percentage of true positives correctly recognized from all positive instances. Precision and Recall are closely related, and they form the foundation for calculating the F1 Score, which balances both metrics.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.5}$$

7. **F1 Score:** The F1 score harmonizes precision and recall, presenting a balanced assessment of model effectiveness. It takes into account both the accuracy in identifying positive instances (precision) and the effectiveness in capturing all positive instances (recall), making it a comprehensive indicator of the model's ability to handle both aspects simultaneously.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.6}$$

8. **False positive rate (FPR):** The proportion of incorrect positive identifications compared to all the negative instances. This shows how often irrelevant items are mistak-

enly recognized as positives.

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.7}$$

9. **False negative rate (FNR):** The ratio of false negatives relative to the total true positives. This indicates how frequently the model overlooks relevant instances.

$$\text{False Negative Rate} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{4.8}$$

4.3 Experiments and results

In this section, we will discuss the experiments we conducted on tuning hyperparameters and exploring different model architectures. During the training process, we assess the model's performance using training and validation loss, with MSE as the chosen loss function.

The training loss is calculated on the training dataset giving an indication how well the model is learning from the training data. It is used to update the model weights through backpropagation.

The validation loss is calculated on the validation dataset, it indicates how well the model generalizes to unknown data. It allows in hyperparameters tunning and preventing overfitting.

4.3.1 Anomaly detection process

The dataset lacked labeled anomalous DC power values and specific guidelines for defining anomalous DC power. Therefore, we have decided to use a synthetic threshold to identify anomalies and evaluate the Transformer model's anomaly detection capability. The chosen threshold was selected based on the behaviour of the DC power. Anomalies were identified by comparing the actual and predicted DC power values against deviations from this threshold. The threshold is set on the normalized DC values as follows: The chosen threshold, set at 0.9 and 0.1 during sunlight hours, was determined for testing purposes without the input of expert opinion.

- **Upper Threshold:** Any DC power value that surpasses the upper threshold of 0.9 is considered an anomaly. This occurrence could suggest an abnormally high generation of DC power.

- **Lower Threshold :** DC power values below the lower threshold of 0.1 are identified as anomalies. These occurrences may indicate power generation dips, system failures, or environmental factors affecting solar panel efficiency. Additionally, inverters can sometimes indicate negative values, which are also considered anomalies.

The chosen synthetic threshold is subject to change and will be updated once real anomalous DC power values and information about the power plant system are accessible. Any adjustments to the threshold and corresponding code will be based on the characteristics of the real dataset.

4.3.2 Hyperparameter tuning

Hyperparameters are settings that guide the learning process, influencing the values of model parameters that a learning algorithm ultimately learns. Tuning these hyperparameters enables us to optimize model performance for the best possible results.

- **Epochs and Batch_size:** We experimented with different values for batch size and the number of epochs to find a balance between underfitting and overfitting. The results are given in Table 4.1, Table 4.2 and Figure 4.2.

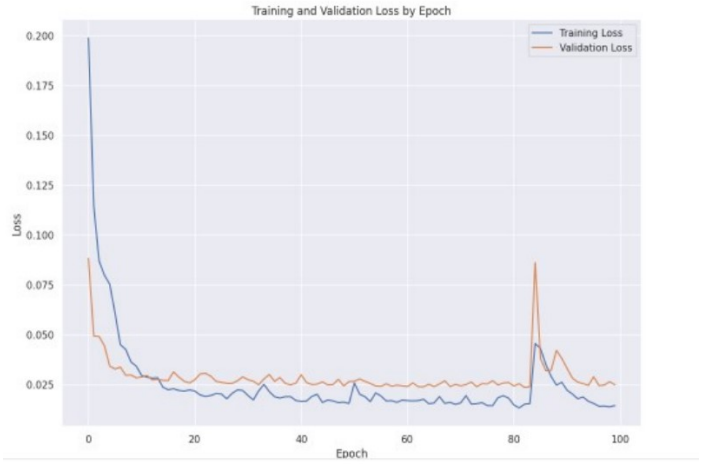
Table 4.1: Effect of epochs on loss.

Epochs	**40**	**43**	**48**	**50**	**100**	**150**	**200**
Train_loss	0.0379	0.0341	0.0196	0.0591	0.0282	0.0131	0.0119
Val_loss	0.0506	0.0301	0.0272	0.0544	0.0347	0.0280	0.0232

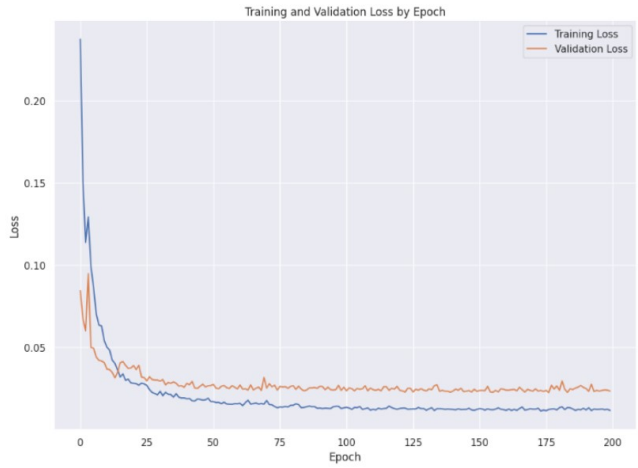
Table 4.2: Effect of batch_size on loss.

Batch_size	**32**	**16**	**42**
Train_loss	0.0132	0.0110	0.0136
Val_Loss	0.0294	0.0266	0.0261

(a) 50



(b) 100



(c) 200

Figure 4.2: Tuning number of epochs.

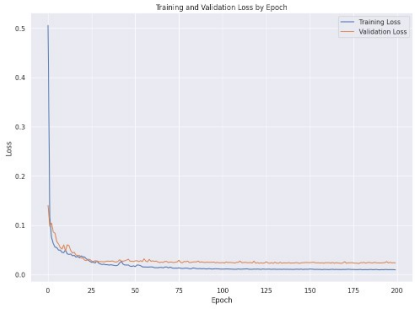
According to the results, epochs=200 and Batch_size = 32, the model appears to perform better in terms of both training and validation losses.

- **MLP units:** This parameter refers to the number of neurons in the hidden layers of the feed forward neural network (FFN) component within the Transformer model. To explore the impact of varying the number of MLP units on code efficiency, we initially set "MLP units" to [128], indicating a single hidden layer in the FFN with 128 neurons. This parameter governs the FFN's capacity and complexity, influencing its ability to discern intricate data patterns.

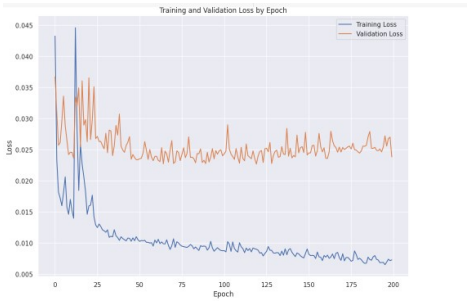
  Increasing the number of units can enhance the model's performance by enabling it to learn more nuanced data relationships. However, this adjustment also heightens the risk of overfitting, particularly when working with smaller datasets. The results are given in Table 4.3 and Figure 4.2.

Table 4.3: Effect of number of MLP Units on loss. a=[64, 32], b=[128, 64, 32] and c=[32, 64, 128].

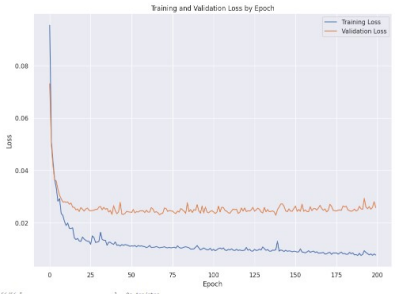
MLP units	**128**	**64**	**32**	**256**	**16**	**512**	**a**	**b**	**c**
Train_Loss	0.0078	0.0094	0.0106	0.0121	0.0105	0.0073	0.0076	0.0051	0.0056
Val_Loss	0.0247	0.0236	0.0236	0.0244	0.0245	0.0238	0.0257	0.0257	0.0267



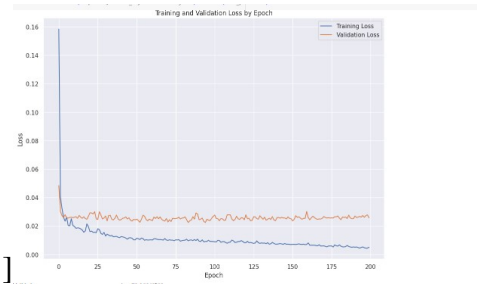
(a) mlp_units=64



(b) mlp_units=512



]
(c)
mlp_units=[64,32



]
(d)
mlp_units=
[128,64,32

Figure 4.3: Tuning MLP units parameter.

We came to the decision of setting `mlp_units=64` as the best result.

- **Optimizer selection :**

  We conducted an experiment comparing five optimizer: Adam, Nadam, Adamax,

RMSprop, and AMSGrad, aiming to determine the best performer. Default learning rates were used for each optimizer.

Table 4.4: Optimizer selection results.

Optimizers	**Nadam**	**Adam**	**Adamax**	**RMSprop**	**AMSGrad**
Training MAE	0.06	0.06	0.06	0.07	0.07
Training MSE	0.01	0.01	0.01	0.01	0.01
Training RMSE	0.09	0.09	0.10	0.10	0.10
Validation MAE	0.09	0.09	0.09	0.10	0.09
Validation MSE	0.02	0.02	0.02	0.02	0.02
Validation RMSE	0.15	0.15	0.15	0.15	0.15
Training Time (seconds)	275.60	247.91	230.36	242.21	269.23
Total Anomalies	431	431	431	431	431
Predicted Anomalies	406	361	348	374	403
True Positives	347	335	325	339	348
False Positives	59	26	23	35	55
True Negatives	1292	1325	1328	1316	1296
False Negatives	84	96	106	92	83
TPR (Recall)	0.81	0.78	0.75	0.79	0.81
FPR	0.04	0.02	0.02	0.03	0.04
Precision	0.85	0.93	0.93	0.91	0.86
F1 Score	0.83	0.85	0.83	0.84	0.83

- **Adam:** Emerges as the overall best performer due to its exceptional precision (0.93) and the highest F1 Score (0.85), demonstrating robust performance across both forecasting and anomaly detection tasks. It maintains a reasonable training time of 247.91 seconds, making it suitable for applications where efficiency and high performance are crucial.

- **Adamax:** Stands out for its shortest training time (230.36 seconds) while maintaining high precision (0.93) in anomaly detection. However, it exhibits lower recall (0.75), impacting its overall F1 Score (0.83), making it ideal for scenarios prioritizing shorter training durations.

- **Nadam and AMSGrad:** Both optimizers offer a balanced approach with high recall (0.81), balanced precision, and similar F1 Scores (0.83). Nadam, despite its longer training time, provides better performance comparable to AMSGrad in terms of precision and recall.

- **RMSprop:** RMSprop demonstrates a balanced approach in anomaly detection with a precision of 0.91, recall of 0.79, and F1 score of 0.84. While its forecasting performance is slightly behind other optimizers, indicated by higher training (MAE: 0.07, RMSE: 0.10) and validation metrics, it maintains moderate training time at 242.21 seconds.

In summary, Adam proves to be the optimal choice when prioritizing precision and balanced performance across forecasting and anomaly detection tasks.

4.3.3 Architecture experiments

- **Head size and number of heads:** head_size represents the dimensionality of each attention head. The attention mechanism in Transformers involves splitting the input into multiple heads, each with its own learned weight matrix. The head_size parameter determines the size of the weight matrix for each head and num_heads represents the number of attention heads used in the model. it is important to find a balance between the **head_size** and **num_heads** parameters that allows the model to capture the necessary patterns in the data while also being computationally efficient.

  Let's discuss how changing the number of heads and their size might affect the accuracy of the model:

Table 4.5: Effect of head_size on loss

Head size	**64**	**32**	**16**	**8**	**128**
Train_loss	0.0142	0.0115	0.0118	0.0139	0.0130
Val_Loss	0.0299	0.0232	0.0273	0.0290	0.0260

Table 4.6: Effect of head_number on loss

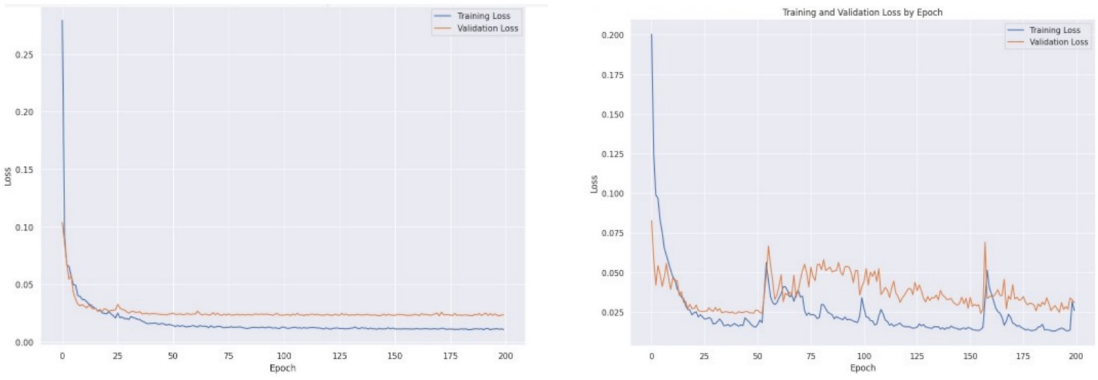
Head number	**6**	**7**	**8**	**9**	**10**
Train_loss	0.0132	0.0117	0.0123	0.0139	0.0133
Val_Loss	0.0294	0.0244	0.0237	0.0261	0.0259

  After rigorous experimentation with head size and number of heads in our Transformer model, we opted for seven heads with a dimensionality of 32. This choice effectively balanced computational efficiency with model performance, yielding robust results in capturing intricate data patterns.

- **Number of Transformer blocks :** This experiment determines the number of Transformer blocks in the model. Each Transformer block consists of two main components: the multi-head attention mechanism and the feed forward neural network. Increasing the number of Transformer blocks could potentially lead to overfitting if the model becomes too complex relative to the size of the training data. Conversely, reducing the number of blocks may limit the model's capacity to learn from the data, potentially leading to underfitting. The results for tuning the number of Transformer blocks are provided in Table 4.7 and Figure 4.4.

Table 4.7: Effect of number of Transformer blocks on loss.

num_Transformer_blocks	**2**	**1**	**3**
Train_Loss	0.0127	0.0108	0.0260
Val_Loss	0.0251	0.0234	0.0309



(a) One block.

(b) Three blocks.

Figure 4.4: Tuning number of Transformer blocks.

Therefore, num_Transformer_blocks = 1 seems to perform better in terms of both training and validation losses.

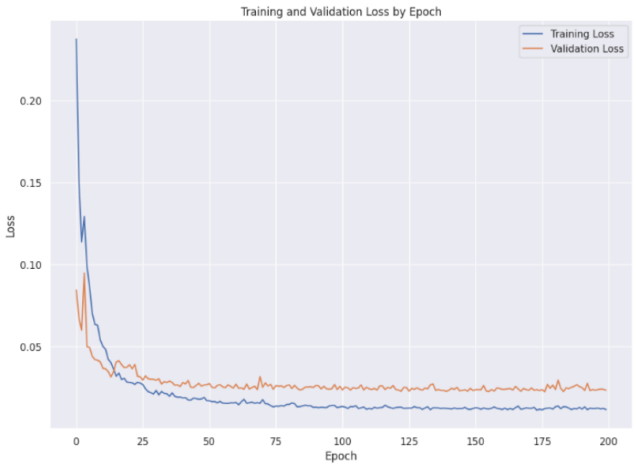
- **Dropout :** Dropout is a regularization technique employed in neural networks to mitigate overfitting by randomly deactivating a portion of input units during training. In our Transformer model, $drop\_out$ governs the dropout rate applied globally, while $drop\_out\_mlp$ is specifically used within the MLP component. This parameter influences dropout regularization after each hidden layer in the MLP, impacting the model's ability to learn representations across different layers. Higher dropout rates in the MLP can enhance generalization by preventing overfitting, albeit potentially extending training times and affecting the model's capacity to capture intricate data relationships. Adjusting $drop\_out$ can also influence the model's generalization ability and training speed, with higher rates aiding in overfitting prevention but possibly slowing down training and limiting the model's ability to learn complex patterns. Results are provide in Table 4.8 and Figure 4.5.

Table 4.8: Effect of drop_out_mlp on loss

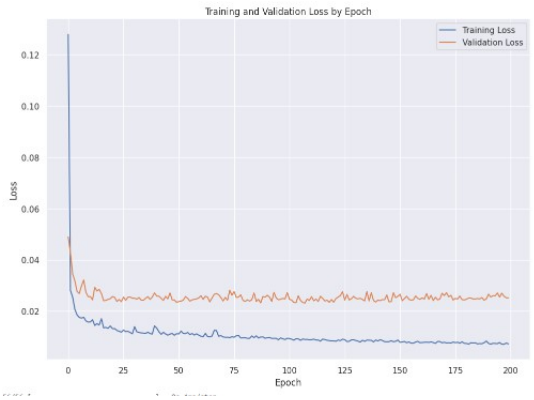
Drop_out_mlp	**0.4**	**0.3**	**0.2**	**0.1**	**0.05**
Train_loss	0.0120	0.0101	0.0103	0.0082	0.0084
Val_Loss	0.0233	0.0227	0.0243	0.0233	0.0248

Table 4.9: Effect of drop_out on loss.

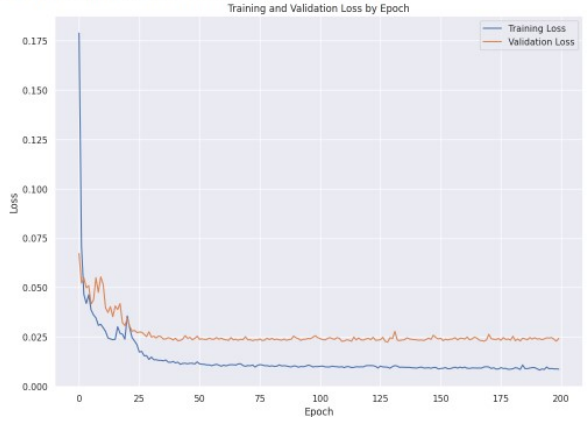
Drop_out	**0.25**	**0.125**	**0.5**	**0.10**
Train_loss	0.0094	0.0074	0.0082	0.0080
Val_Loss	0.0237	0.0240	0.0233	0.0244



(a) 0.25



(b) 0.10



(c) 0.05

Figure 4.5: Tuning MLP Drop out parameter.

Based on the experiments conducted, the optimal dropout settings are identified as:

- $drop\_out\_mlp = 0.025$
- $drop\_out = 0.25$

These configurations were found to minimize both training and validation losses ef-

fectively, demonstrating robust control over overfitting within the model.

- **Encoder architecture:**

  The experiments were carried out on the encoder with a feed-forward dimension of 8, which provided the best performance for LSTM with a validation loss of 0.0235 when compared to 1DCNN and GRU. These experiments investigated several encoder architectures, including those with and without positional encoding. Within each version, we examined various types of feed-forward blocks, including LSTM, GRU, and 1D CNN layers. the table 4.10 presents a concise overview of the experimental outcomes.

Table 4.10: Effect of positional encoding on loss.

Without positional encoding				**With positional encoding**			
Metric	**LSTM layers**	**1DCNN layers**	**GRU layers**	**LSTM layers**	**1DCNN layers**	**GRU layers**	**Hybrid layer (LSTM & 1DCNN)**
Train_loss	0.0089	0.0090	0.0096	0.0092	0.0091	0.0095	0.0090
Val_loss	0.0241	0.0260	0.0265	0.0225	0.0259	0.0262	0.0340
Train_MAE	0.06	0.06	0.063	0.06	0.061	0.063	0.059
Val_MAE	0.09	0.09	0.093	0.09	0.091	0.093	0.089

The results in table 4.10 showed that while the LSTM layers without positional encoding had the lowest training loss (0.0089), the inclusion of positional encoding led to the lowest validation loss (0.0225) compared to the 1DCNN and GRU, indicating better generalization and performance on unseen data. Additionally, the LSTM layers consistently exhibited low training and validation MAE, with or without positional encoding. Despite a slight increase in training loss with positional encoding, the significant improvement in validation loss justified its selection, demonstrating that implementing PE with LSTM layers effectively balances training and validation performance. Even the hybrid layer, LSTM and 1DCNN, did not give better results than the pure LSTM layers.

- **Incorporating Kolmogorov-Arnold Network:**

  This experiment aims to enhance our Transformer-based model by comparing the conventional MLP layer with the advanced KAN. It also explores various hybrid configurations to optimize performance while retaining the core architecture and attention mechanisms of the Transformer model. The study evaluates seven different configuration.

  Configurations Tested:

  1. **Initial configuration**: A dense layer followed by a dropout layer.

  2. **KAN replacement**: Replace the dense layer with one layer of KAN. During this phase, hyperparameter tuning was performed. Initially, different unit values were experimented with: 16, 32, 64, 128, and 512. The best results were achieved with 64 units. Subsequently, the number of epochs and batch size were adjusted, but these changes did not lead to any improvements.

3. **Hybrid configuration**: A dense layer followed by a KAN layer and a dropout layer.

4. **Reversed configuration**: A KAN layer followed by a dropout layer and then a dense layer.

5. **Alternative reversed configuration**: A KAN layer followed by a dense layer and then a dropout layer.

6. **Two layers of KAN**: Two layers of KAN followed by a dropout layer.

7. **Complex layer sequence**: Two layers of KAN followed by a dense layer and then a dropout layer.

The optimal model configuration was selected based on its performance in both forecasting accuracy and anomaly detection capability. Results are summarized in the table below:

Table 4.11: Effect of KAN on anomaly detection performance.

Configurations	Total predicted anomalies	TP	FP	TR	FN	TPR	FPR	Precision	Recall	F1 score	Total trainable parameters
Initial config	381	341	40	1311	90	0.79	0.03	0.90	0.79	0.84	6533
KAN replacement	300	283	17	1334	148	0.66	0.01	0.94	0.66	0.77	6469
Hybrid config	354	314	40	1311	117	0.73	0.03	0.89	0.73	0.8	10629
Reversed config	395	349	46	1305	82	0.81	0.03	0.88	0.81	0.8	10629
Alternative reversed config	252	241	11	1340	190	0.56	0.01	0.96	0.56	0.7	10565
Two layers of KAN	361	336	25	1326	95	0.78	0.02	0.93	0.78	0.85	10629
Complex layer sequence	411	354	20	1313	77	0.82	0.01	0.94	0.82	0.87	14725

Table 4.12: Effect of KAN on anomaly forecasting performance.

Configurations	Training MAE	Training MSE	Training RMSE	Validation MAE	Validation MSE	Validation RMSE
Initial configuration	0.07	0.15	0.38	0.09	0.13	0.36
KAN replacement	0.06	0.01	0.09	0.09	0.02	0.15
Hybrid configuration	0.06	0.15	0.39	0.09	0.13	0.36
Reversed configuration	0.06	0.16	0.40	0.10	0.14	0.37
Alternative reversed configuration	0.06	0.15	0.38	0.09	0.13	0.36
Two Layers of KAN	0.07	0.15	0.39	0.10	0.13	0.36
Complex layer sequence	0.06	0.01	0.09	0.09	0.02	0.15

The performance metrics from the experiments indicate that both the alternative reversed configuration and the complex layer sequence performed well. However, the complex layer sequence, which consists of two KAN layers followed by a dense layer and a dropout layer, showed a slightly better balance between precision, recall, and F1 score.

This experiment highlights the synergy of traditional MLP layers and KAN layers in Transformer-based model for time series forecasting and anomaly detection. The Complex Layer Sequence emerged as the top performer, balancing enhanced forecasting accuracy and improved robustness in anomaly detection. It matches the ”Dense only” setup in forecasting accuracy while surpassing it in overall model robustness and generalization, making it ideal for both tasks.

4.4 Comparison study analysis

We conducted a comparative study between our Transformer model and LSTM model for time series forecasting of solar power predictions, we evaluated their performance across several metrics with a focus on anomaly detection and forecasting accuracy.

4.4.1 Forecasting performance analysis

The table 4.13 compares the performance of the Transformer and LSTM models in forecasting task:

Table 4.13: Forecasting performance.

Model	**Train_loss**	**Val_loss**	**Training time** (seconds)
Transformer	0.0087	0.0243	310.87
LSTM	0.0099	0.0247	84.31

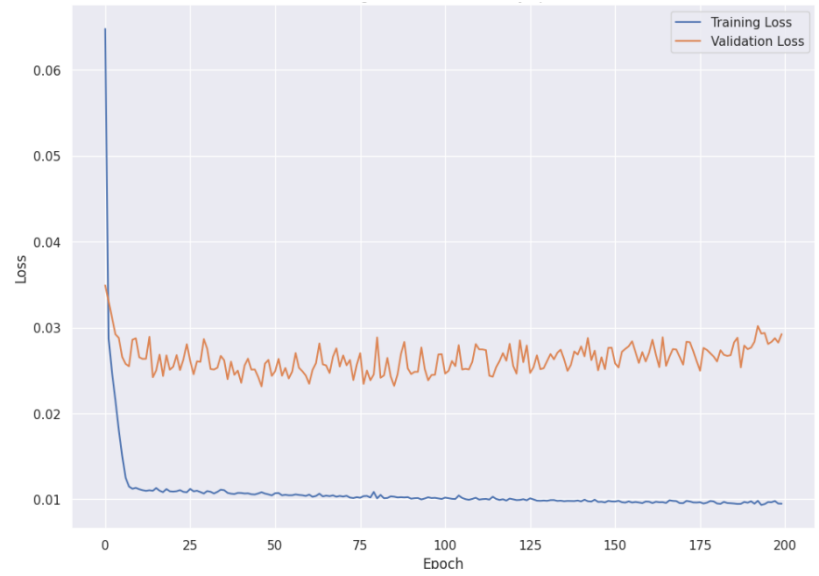
- The Transformer model exhibits a lower training loss (0.0087) compared to the LSTM model (0.0099), indicating better training performance. Similarly, the validation loss for the Transformer (0.0243) is marginally lower than that of the LSTM (0.0247), suggesting slightly better generalization to unseen data.

- The LSTM model shows higher training (0.0099) and validation losses (0.0247) than the Transformer, indicating slightly worse performance in both training and validation phases.

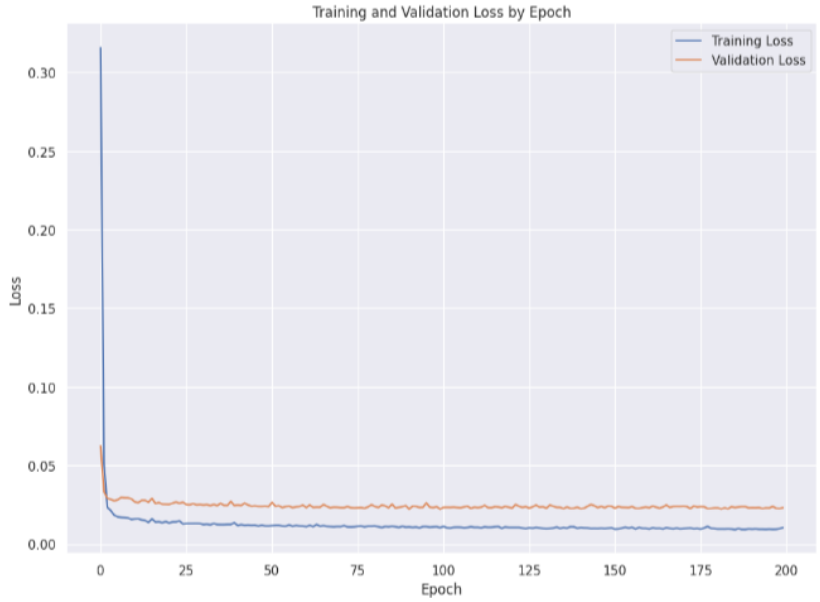
- The training time for the Transformer model is significantly longer, totaling 310.87 seconds, in contrast to the LSTM model's training time of 84.31 seconds.

The Transformer model exhibits improved loss metrics compared to the LSTM model, indicating enhanced predictive precision. However, this enhanced performance is accompanied by a substantially longer training duration owing to the Transformer model's heightened computational complexity. Nevertheless, in our specific context where computational resources are not a primary constraint, the extended training time of the Transformer model is negligible.

The figures below shows the distribution of training loss and validation loss for each model:



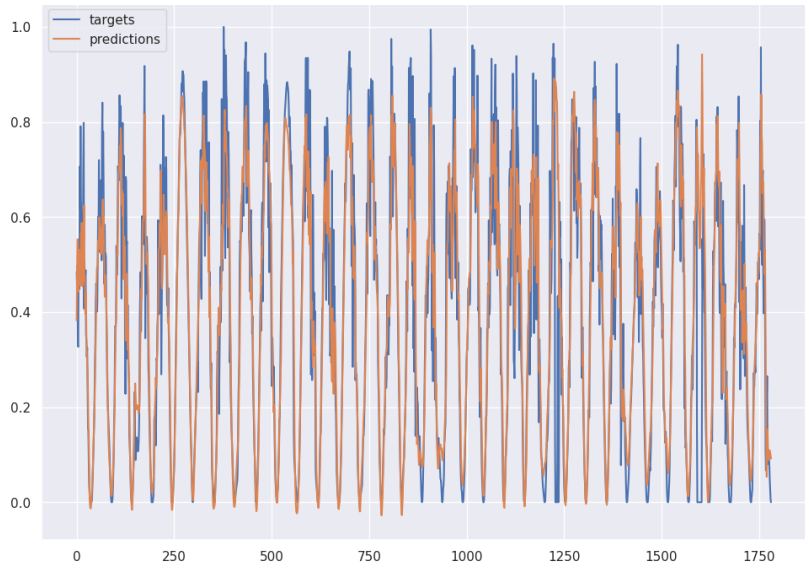
(a) LSTM



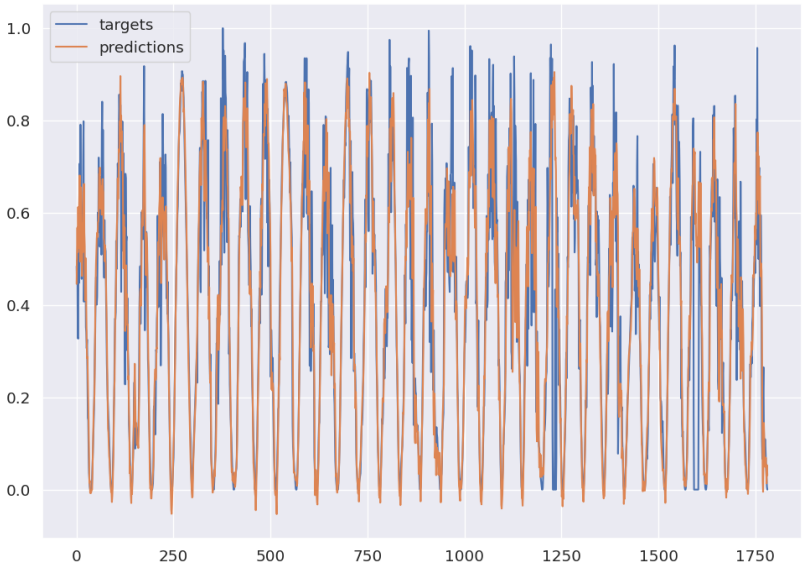
(b) Transformer

Figure 4.6: Loss over epochs for both models.

Figure 4.7 shows the forecast of each model:



(a) LSTM



(b) Transformer

Figure 4.7: Forecasting results.

4.4.2 Anomaly detection performance analysis

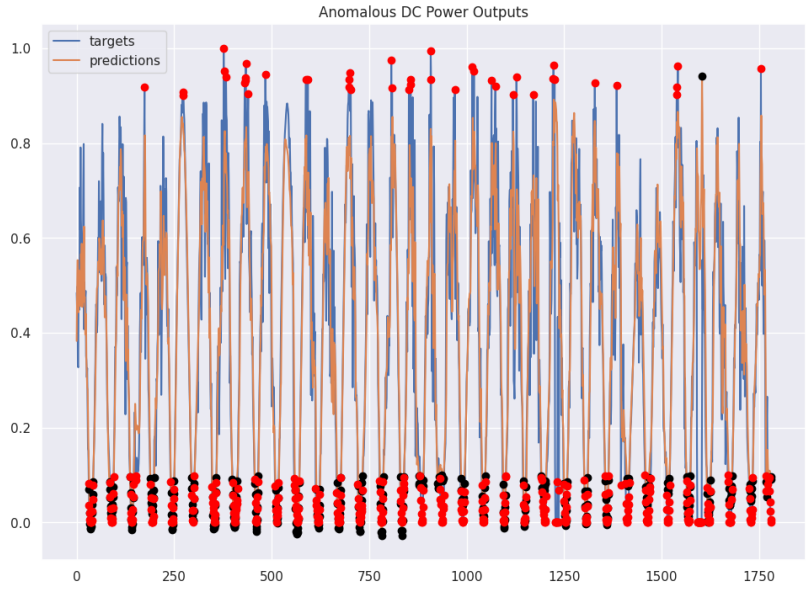
The table 4.14 compares the performance of the Transformer and LSTM models in detecting anomalies.

Table 4.14: Anomaly detection performance.

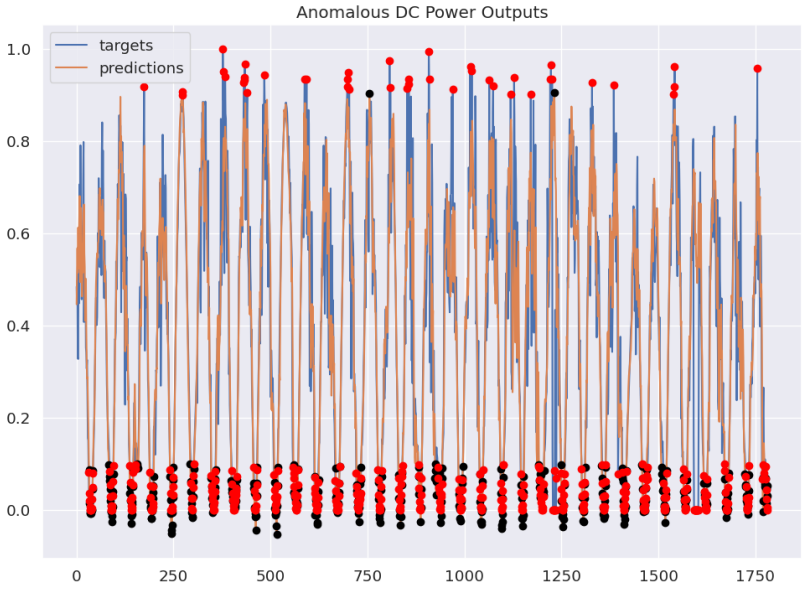
Model	Total anomalies	Predicted anomalies	TP	FP	TN	FN	TPR	FPR	Precision	Recall	F1 score
Transformer	431	411	354	20	1313	77	0.82	0.01	94%	0.82	0.87
LSTM	431	325	305	38	1331	116	0.72	0.02	89%	0.72	0.79

- The Transformer model outperforms the LSTM model in terms of recall, detecting a higher proportion of true anomalies (82% vs. 72%). This indicates that the Transformer is more effective in identifying actual anomalies.
- The Transformer's higher precision (94% vs. 89%) means it has a lower rate of false positives, making it more reliable in anomaly detection.
- The Transformer's higher F1 score (0.87 vs. 0.79) demonstrates its superior balance between precision and recall, making it more effective overall.
- Both models have a low false positive rate, but the Transformer's FPR (0.01) is slightly better than the LSTM's (0.02), indicating fewer false alarms.

The figure 4.8 shows the detection of anomalies for both models,the actual anomalies are shown in red and the predicted anomalies in black:



(a) LSTM



(b) Transformer

Figure 4.8: Distribution of anomaly detection results.

The Transformer model demonstrates better anomaly detection performance than the LSTM model, exhibiting higher recall, precision, and F1 score. These metrics confirm that the Transformer model is more precise and dependable in detecting anomalies. In contrast, the LSTM model struggles to maintain a balance between precision and recall, resulting in a higher number of missed anomalies and slightly more false positives. Thus, for anomaly detection tasks within this dataset, the Transformer model emerges as the more resilient and effective option.

4.5 Conclusion

In this chapter, we documented the evolution of our initial Transformer architecture, which was rigorously tested against LSTM model. Our findings firmly establish that the Transformer approach excels in solar power generation forecasting and anomaly detection, consistently outperforming traditional LSTM architecture. The robustness of our refined Transformer, developed through iterative improvements and comparative evaluations, underscores its superiority. This chapter resoundingly reaffirms the Transformer's effectiveness in leveraging temporal dependencies and global contextual information, establishing it as the optimal choice for precise and reliable solar power forecasting and anomaly detection in our research

Conclusion

This project has explored the transformative potential of Transformer-based models for solar power production and its potential in time series forecasting for solar power prediction. Through meticulous experimentation and comparative analysis, we have demonstrated the remarkable advancements made possible by leveraging advanced Transformer architectures.

Our findings firmly establish that the Transformer approach excels in solar power generation forecasting and anomaly detection, particularly when the Kolmogorov-Arnold Network is incorporated into the architecture. Indeed, the Transformer model consistently outperforms the LSTM architecture, showcasing remarkable efficiency. The robustness of our refined Transformer, developed through iterative improvements and comparative evaluations, underscores its superiority. This reaffirms the Transformer's effectiveness in leveraging temporal dependencies and global contextual information, making it the optimal choice for precise and reliable solar power forecasting and anomaly detection in our research. Besides, our study highlighted the importance of advancing AI integration in the solar energy sector, particularly in leveraging the dual capabilities of time series forecasting and anomaly prediction to enhance the efficiency of solar power systems outcome. In this context, our research contributes paving the way for smarter and more efficient energy solutions in the era of renewable resources.

Our vision is to deploy the Transformer model in real-world settings, in our country context, integrating it with existing infrastructure and incorporating real-time data labeled with anomalous values. This would enable the model to learn from real-world data and adapt to changing conditions, ultimately enhancing its accuracy and reliability. Additionally, incorporating real-time data would allow for more effective anomaly detection and forecasting, enabling proactive maintenance and grid management strategies. To achieve this, we aim to collaborate with industry partners to deploy the model in real-world settings, such as solar power plants or grid management systems.

This integration could benefit from edge AI concepts, bringing computation closer to the data source, reducing latency, and enabling rapid decision-making in remote and distributed solar installations. This would involve addressing challenges related to scalability, robustness, and integration with existing infrastructure. By tackling these challenges and incorporating real-time data, the Transformer model can be further refined and optimized for real-world applications, ultimately contributing to the more efficient and reliable integration of renewable energy sources into the global energy mix. Additionally, we plan to explore different architectures and hybrid techniques on the Transformer and conduct more

comparative studies with other existing solutions.

Bibliography

[1] C. Bacher, M. Möller, and Y. Schön. "Solar radiation forecasting for the power grid: A review". In: *Renew. Sustain. Energy Rev.* 13 (2009), pp. 1456–1465. DOI: 10.1016/j.rser.2009.02.004.

[2] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *ArXiv* abs/1412.3555 (2014). URL: https://api.semanticscholar.org/CorpusID:5201925.

[3] M Elsaraiti and A Merabet. "Solar Power Forecasting Using Deep Learning Techniques". In: *IEEE Access* 10 (2022), pp. 31692–31698.

[4] Freepik. *Solar Power Generation*. February 2024. URL: %7Bhttps://www.freepik.com/search?format=search&last_filter=query&last_value=solar+power+generation&query=solar+power+generation%7D.

[5] et al. Huang. "Deep Learning Global Horizontal Irradiance Prediction (DL-GHIP) model for Solar Power Prediction". In: (2019). URL: https://www.example.com/dl-ghip-model.

[6] Shaohan Huang et al. "HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log". In: *IEEE Transactions on Network and Service Management* 17.4 (2020), pp. 2064–2076. DOI: 10.1109/TNSM.2020.3034647. URL: https://doi.org/10.1109/TNSM.2020.3034647.

[7] M. Husein and I.-Y. Chung. "Day-ahead solar irradiance forecasting for microgrids using a long short-term memory recurrent neural network: A deep learning approach". In: *Energies* 12 (2019), p. 1856. DOI: 10.3390/en12151856. URL: https://www.mdpi.com/1996-1073/12/10/1856.

[8] Edmond LEZMI Jiali XU. "Time Series Forecasting with Transformer Models and Application to Asset Management". In: *Amundi Institute Publications.* (Feb. 2023). Available at https://papers.ssrn.com/sol3/papers.cfm?abstract$_id = 4375798$.

[9] Chenen Jin. ""Application and Optimization of Long Short-term Memory in Time Series Forcasting"". In: *2022 International Communication Engineering and Cloud Computing Conference (CECCC)*. 2022. DOI: 10.1109/CECCC56460.2022.10069825.

[10] Anni Kanam. *Solar Power Generation Data*. Solar power generation and sensor data for two power plants. 2020. URL: https://www.kaggle.com/datasets/anikannal/solar-power-generation-data.

[11] et al. Li. "Recurrent Neural Networks Based Photovoltaic Power Forecasting Approach". In: *Energies* 12 (2019), p. 2538. DOI: 10.3390/en12132538.

[12] B. Lim, T. L. Lai, and P. S. Loh. "Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting". In: *arXiv preprint arXiv:1910.13101* (2019). URL: https://arxiv.org/abs/1910.13101.

[13] Ziming Liu et al. "KAN: Kolmogorov-Arnold Networks". In: 2024. URL: https://api.semanticscholar.org/CorpusID:269457619.

[14] J. Lorentz et al. "Forecasting solar irradiance using numerical weather prediction models". In: *Journal of Physics: Conference Series* 1347 (2022), pp. 1347–1355. DOI: 10.35848/1347-4065/acd9b5.

[15] Mahyar Amini Melika Heydari Ashkan Heydari. "Solar Power Generation and Sustainable Energy: A Review". In: *international Journal of Technology and Scientific Research,* 12 (2023). https://doi.org/10.1038/s41598-023-46735-3.

[16] B. N. Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *arXiv* arXiv:2003.04887 (2020).

[17] J. Oruh, S. Viriri, and A. Adegun. "Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition". In: *IEEE Access* 10 (2022), pp. 30069–30079.

[18] Marco Peixeiro. *Time Series Forecasting in Python*. 1st. Vol. 1. Shelter Island, NY 11964: Manning Publications Co, 2022.

[19] George Athanasopoulos Rob J Hyndman. *Forecasting: Principles and Practice*. 2nd edition. Available at OTexts.com/fpp2. :Melbourne, Australia: OTexts, (2018).

[20] Brenda Rose. *How is the electricity generated by a solar power plant distributed to consumers?* 2023. URL: https://medium.com/@Breadarose/how-is-the-electricity-generated-by-a-solar-power-plant-distributed-to-consumers-b684c3c304f3.

[21] H. Sak, A. W. Senior, and F. Beaufays. "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling". In: *ArXiv e-prints* (Feb. 2014). DOI: 10.1007/978-981-13-1595-4_4.

[22] D. Salman et al. "Hybrid deep learning models for time series forecasting of solar power". In: *Neural Comput & Applic* 36 (Feb. 2024), pp. 9095–9112. URL: https://doi.org/10.1007/s00521-024-09558-5.

[23] Sharma et al. "A review of solar forecasting techniques and the role of artificial intelligence". In: *Solar* 4 (2021), pp. 99–135. DOI: 10.3390/solar4010005.

[24] Yash Akash Singh. "CNN Model for Time Series Analysis". In: *Medium* (2021). URL: https://medium.com/@yashakash.singh7/cnn-model-for-time-series-analysis-3b58b4254790.

[25] S. Sobri et al. "Solar photovoltaic generation forecasting methods: A review". In: *Energy Convers. Manag.* 156 (2021), pp. 459–497. DOI: 10.1016/j.enconman.2017.12.034.

[26] William Holderbaum Stephen Haben Marcus Voss. *Core Concepts and Methods in Load Forecasting with Applications in Distribution Networks*. 1st. Available at https://doi.org/10.1007/978-3-031-27852-5. Gewerbestrasse 11, 6330 Cham, Switzerland: Springer Cham, 1 May 2023.

[27] et al. Suresh. "Solar Irradiance Forecasting Using Deep Learning Techniques". In: *Research Repository* (2020).

[28] Yi Tay et al. "Efficient Transformers: A Survey". In: *ACM Digital Library* (Dec. 2022). arXiv:2009.06732v3 [cs.LG], 14 Mar 2022. DOI: 10.1145/3530811. URL: https://dl.acm.org/doi/10.1145/3530811.

[29] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. "Tranad: Deep transformer networks for anomaly detection in multivariate time series data". In: *arXiv preprint arXiv:2201.07284* (2022).

[30] Unknown. *Architecture of LSTM*. https://www.researchgate.net/figure/architecture-of-LSTM_fig2_368953502. Unknown Unknown.

[31] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[32] et al. Wang. "Solar Transformer: A Novel Method for Solar Irradiance Forecasting". In: *Energies* 16 (2022), p. 7610. DOI: 10.3390/en16227610. URL: https://www.mdpi.com/1996-1073/16/22/7610.

[33] Na Wang and Xianglian Zhao. "A decoder-only foundation model for time-series forecasting". In: *arXiv* (2023).

[34] Na Wang and Xianglian Zhao. "Enformer: Encoder-Based Sparse Periodic Self-Attention Time-Series Forecasting". In: *IEEE Access* 11 (2023), pp. 112004–112014. DOI: 10.1109/ACCESS.2023.3322957.

[35] Wen et al. "Congestion control strategies for increased renewable penetration of photovoltaic in LV distribution networks". In: *Energy Rep.* 8 (2019), pp. 217–223. DOI: 10.1016/j.egyr.2019.02.012.

[36] Qingsong Wen et al. "Transformers in Time Series: A Survey". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[37] Y. Zhang, J. Zhang, and X. Zhang. "Forecasting Solar Irradiance with Long Short-Term Memory Recurrent Neural Networks". In: *Energy* 221 (2021), p. 119887. DOI: 10.1016/j.energy.2021.119887.

[38] Y. Zhang, J. Zhang, and X. Zhang. "Solar radiation estimation in different climates with meteorological variables using Bayesian model averaging and new soft computing models". In: *Renewable Energy* 169 (2021), pp. 1171–1183. DOI: 10.1016/j.renene.2021.02.024.

[39] J. Zhu et al. "Time-Series Power Forecasting for Wind and Solar Energy Based on the SL-Transformer". In: *MDPI* (2021). DOI: 10.3390/en14072045. URL: https://www.mdpi.com/1996-1073/14/7/2045.

[40] Zoumana. "An Introduction to Convolutional Neural Networks (CNNs)". In: *DataCamp* (2021). https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns.