Olukayode AKANNI

OPTIMIZING ENERGY IN THE DIGITAL AGE TRANSFORMER FOR SOLAR IRRADIANCE FORECASTING

MASTERS THESIS

2023

NEAR EAST UNIVERSITY

INSTITUTE OF GRADUATE STUDIES

ARTIFICIAL INTELLIGENCE ENGINEERING DEPARTMENT

OPTIMIZING ENERGY IN THE DIGITAL AGE: TRANSFORMER FOR SOLAR IRRADIANCE FORECASTING

M.Sc. THESIS

Olukayode AKANNI

Nicosia

June 2023

NEAR EAST UNIVERSITY
INSTITUTE OF GRADUATE STUDIES
ARTIFICIAL INTELLIGENCE ENGINEERING DEPARTMENT

OPTIMIZING ENERGY IN THE DIGITAL AGE TRANSFORMER FOR SOLAR IRRADIANCE FORECASTING

M.Sc. THESIS

Olukayode AKANNI

Supervisor
Prof. Dr. Fadi AL-Turjman

Nicosia
June 2023

Approval

We certify that we have read the thesis submitted by **Olukayode AKANNI** titled **"Optimizing Energy In The Digital Age: Transformer For Solar Irradiance Forecasting"** and that in our combined opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Examining Committee	Name- Surname	Signature
Head of the Committee:	Assoc. Prof. Dr. Sertan Serte	....
Committee Member:	Asst. Prof Dr. Pwadubashiyi Coston	....
Supervisor:	Prof. Dr. Fadi Al-Turjman	....

Approved by the head of the Department

26/06/2023

....

Prof. Dr. Fadi Al-Turjman

Head of Department

Approved by the Institute of Graduate Studies

....../2023

Prof. Dr. Kemal Hüsnü Can Başer

Head of the Institute

Declaration

I hereby declare that all information, documents, analysis and results in this thesis have been collected and presented according to the academic rules and ethical guidelines of Institute of Graduate studies, Near East University. I also declare that as required by these rules and conduct, I have fully cited and referenced information and data that are not original to this study.

Olukayode AKANNI

04./10./2023

Acknowledgments

I write to express my profound thanks to Prof. Dr. Fadi AL-TURJMAN, for his supervision, tolerance, as well as leadership all through my postgraduate programme at Near East University. In order to project my long-term professional aspirations, his guidance was crucial in giving a well-rounded experience. He pushed me to follow through and to have confidence in whatever I did, always asking for updates on projects for both this work and my Master's studies. Prof. Dr. Fadi AL-Turjman, I really appreciate everything you have done for me. Finally, I thank my family for unceasing supports, prayers and love received from them at the time I was unavailable. I also acknowledge the support of Dr. Auwalu Saleh Mubarak and Mercel with my heart full of gratitude.

I am forever grateful my loving family; wife-Kemi and children; David, Divine, Daniel, God’s Delight and God’s Power, who provided Daddy the conducive environment to work on this study, and whose constant affection, support and prayers have helped me in pulling through tough circumstances and achieve academic excellence in spite of it.

I also love to extend my appreciation to my course mates, colleagues and loved ones inside and outside the university for being there and making the academic journey a lot easier.

My gratitude goes to the administrative secretary Mrs Sinem AL-Turjman, she helped me solved the puzzles working out an optimum combination of my coursework and recording of course work results in the school’s system.

Finally, I like to thank the Near East University management for giving me scholarship which made my pursuit of this post graduate study a reality.

Olukayode AKANNI

Dedication

The work is totally dedicated to the Almighty God. I thank Jesus, for being the source of my motivation, guidance, financial, emotional, spiritual support and He also gave me courage when I wanted to give up. My appreciation goes to my teachers, coaches, course mates who offered their advice and support to complete this thesis. Finally, my gratitude goes to myself for having the mental capacity, writing abilities necessary to finish this work.

Abstract

"Optimizing Energy In The Digital Age: Transformer For Solar Irradiance Forecasting"

Olukayode AKANNI

M.Sc , Department of Artificial Intelliegnece Engineering

June 2023, 97 pages

Artificial intelligence and renewable energy are critical for attaining a carbon-neutral economy and a sustainable environment.

With rising energy demand, reliable sun irradiance estimates are required for integrating solar photovoltaic (PV) systems into the national power grid.

This research introduces innovation of teacher forcing concept to Time Series Transformer model with attention mechanism for forecasting solar irradiance, similar to the GPT's training framework. The NASA Kaggle dataset was utilized, and it includes meteorological and solar radiation data from September through December of 2016. Temperature, pressure, humidity, radiation, wind direction, sunrise and sunset periods, and time-related variables were extracted from the dataset using Exploratory Data Analysis.

Before inputing the solar radiation data into the Transformer model, it was pre-processed and normalized. Test results show the superiority of the proposed model when compared with the other 10 AI models. The Time Series Transformer model is effective and has the highest performance attained by having the lowest MSE, RMSE, MAE, and R2. When compared to other state-of-the-art MAPE solar forecasting findings, the Time Series Transformer model has 97.6% as coefficient of determination and the lowest Mean Absolute Percentage Error of 0.68%, making it an excellent approach for forecasting solar energy. In the digital era, this model is a helpful tool for energy optimization. A Proof-of Concept implementation of this project can be found here.

Keywords: Solar Irradiance, Transformer Model, Machine learning, Sustainable climate, Artificial Intelligence, Renewable energy, NASA

ÖZET

“Dijital Çağda Enerjiyi Optimize Etmek: Güneş Işınım Tahmini İçin Transformatör”

Olukayode AKANNI
Yüksek Lisans , Yapay Zeka Mühendisliği Bölümü
Haziran 2023, 97 sayfa

Yapay zeka ve yenilenebilir enerji, karbon-nötr bir ekonomi ve sürdürülebilir bir çevre elde etmek için kritik öneme sahiptir. Artan enerji talebi ile birlikte, güvenilir güneş ışınımı tahminleri, güneş fotovoltaik (PV) sistemlerinin milli enerji şebekesine entegrasyonu için gereklidir. Bu araştırma, güneş ışınımını tahmin etmek için Zaman Serisi Transformer modeline öğretmen zorlaması kavramının yenilikçi bir şekilde uygulanmasını sunmaktadır ve GPT'nin eğitim yapısı ile benzerlik gösterir. NASA Kaggle veri seti kullanılarak, Eylül ayından Aralık 2016'ya kadar olan dönemi kapsayan meteorolojik ve güneş radyasyonu verileri elde edildi. Sıcaklık, basınç, nem, radyasyon, rüzgar yönü, gün doğumu ve gün batımı süreleri ile zamanla ilişkili değişkenler, veri keşfi yöntemleri kullanılarak veri setinden çıkarıldı.

Güneş radyasyonu verileri, Zaman Serisi Transformer modeline girmeden önce, önceden işlendi ve normalize edildi. Test sonuçları, önerilen modelin diğer 10 yapay zeka modelleri ile karşılaştırıldığında üstünlüğünü göstermektedir. Zaman Serisi Transformer modeli etkili ve en düşük MSE, RMSE, MAE ve R2 değerlerine sahip olarak en yüksek performansı sergilemektedir. Diğer güncel MAPE güneş tahmin bulgularıyla karşılaştırıldığında, Zaman Serisi Transformer modeli %97,6 belirleme katsayısı ve %0,68'lik en düşük Mutlak Yüzde Hata ile mükemmel bir güneş enerjisi tahmini yaklaşımı sunmaktadır. Dijital çağda bu model, enerji optimizasyonu için faydalı bir araçtır. Bu projenin bir Konsept Kanıtı uygulaması burada bulunabilir.

Anahtar kelimeler: Transformer Model, Makine öğrenmesi, Regresyon, Yapay Zeka, Yenilenebilir enerji, NASA, Regresyon.

Table of Contents

CHAPTER I

CHAPTER II

CHAPTER III

CHAPTER IV

CHAPTER V

List of Appendices

List of Figures

List of Tables

List of Abbreviations

Acronym	Meaning
Covet:	Convolution Neural Network
ANN:	Artificial Neural Network
MSE:	Mean squared error
MAE:	Mean Absolute error
R-square or R^2:	Pearson correlation
AI	Artificial Intelligence
ML:	Machine learning
DL:	Deep learning
CPU:	Computer Processing Unit
GPU:	Graphical Processing Unit
KNN:	K-nearest neighbour (KNN),
MLP:	MLP multilayer perceptron (MLP)
RFR:	Random forest repressor (RFR), and
ARIMA	Auto-regressive integrated moving average (ARIMA)
DL	Deep learning models
CNN:	Convolutional neural networks (CNN)
RNN:	Recurrent neural networks (RNN)
SDGs:	Sustainable Development Goals (SDGs)
LSTM:	Long and short Term Memory
UN:	United Nations
NWP:	Numerical weather prediction (NWP)
EU:	European Union
Iota:	Internet of Things

CHAPTER I
Introduction

1.0 Background of the Study

Our population worldwide has been on the rise for the past decades, which resulted in an increase in the basic needs of the everyday lives of humans (Kumara and Toshniwal, 2021b). Energy tops the lists of resources that must be increased, with an estimated increase of electricity demand expected to reach 70% from 2015 in the next couple of years (Duffy et al., 2015). Solar energy is a popular renewable energy source, making it essential to accurately forecast solar energy production and adjust energy demands accordingly especially with fossil fuels impacting negatively on the environment. The use of solar energy, in particular, has grown significantly. However, solar energy is an intermittent source of energy, and its availability depends on weather conditions. The world has been relying for the past century on fossil fuels for power generation that are not only depletable but also suffer from heavy environmental drawbacks (Kumari and Toshniwal, 2021a) .AI with IOT are crucial for increasing the use of renewable energy, which is critical for reducing greenhouse gas emission that drive climate change, and to monitor energy production.

As a response to that, many countries have been recently investing in renewable energy. Solar energy in particular has been deemed the most promising source due to the abundancy of solar radiation (Wang et al., 2020). Solar Photovoltaics (PV), in particular, have been gaining attention because of their environmental and economic benefits. Its working principle is based on converting the sunlight irradiance into electricity through the photovoltaic effect (Sampaio and González, 2017). Although PV energy offers itself as a cheap and eco-friendly alternative to traditional thermal sources, the integration of PV into the national grid suffers from several drawbacks. PV, like other renewable sources, is intermittent by nature (Kumari and Toshniwal, 2021b). In other words, solar energy production depends on weather factors that vary with time, resulting in a very chaotic and uncontrollable energy output (Brahma and Wadhvani, 2020). When integrated with the electricity grid on a large-scale, PV systems may cause reliability issues due to underproduction, and excessive costs during overproduction, and may consequently degrade the grid (Abdel-Nasser et al., 2020).

For a reliable and economic integration of PV, grid operators must continuously receive accurate forecasts of solar irradiance in real-time (Kumari and Toshniwal, 2021a). Forecasting solar irradiance is essential for optimizing solar panel energy production and incorporating solar energy into the electrical grid. Energy businesses must accurately predict solar irradiance in order to efficiently build and run solar power projects.

Since accurate forecasting methods have been developed for these factors, models have been developed to deduce the irradiance from those forecasts (Lai et al., 2020).

Numerical weather prediction (NWP) is what most of the models developed for solar irradiance forecasting use (Murata et al., 2018). Although they are widely accepted as a decent forecasting technique, they are computationally expensive and require the processing of large datasets (Hao et al., 2019). Consequently, they fail in the case of short-term forecasting needed by energy control centers. Some statistical methods that use regression and time-series techniques have also been utilized. However, their success has been constrained by the non-stationary and non-linear solar irradiation. (Reikard, 2009).

Artificial neural networks (ANN) have emerged recently in the area of machine learning as a successful forecasting model (Kumari and Toshniwal, 2021b; Kumari and Toshniwal, 2021a; Brahma and Wadhvani, 2020; Abdel-Nasser et al., 2020; Wang et al., 2018; Huang et al.,2021).

A number of techniques (Curceac et al., 2019) have been utilized for solar forecasting and prediction, including statistical models, ML models, and deep learning (DL) models. LSTM variations have also lately become the most often used option for time series data modelling. (Middya and Roy, 2022).

The transformer is the cornerstone of modern AI technology. Transformers are a type of deep learning model design, much as CNNs and LSTMs. The benefits of this ground-breaking architecture have prompted the use of Transformers as the basis for the newest cutting-edge models. The Transformer's capacity to examine input simultaneously utilizing many heads of self-attention helps speed up training. The self-attention mechanism, which greatly improves prediction accuracy, also gives the Transformer a larger capacity for data classification/ regression. As of right now, the Transformer has generated noteworthy outcomes in NLP and CV fields (Vaswani et al., 2017; Tetko et al., 2020; Acheampong et al., 2021; Li et al., 2021). The field of NLP has undergone a revolution because of the usage of transformer models., and their potential for time series forecasting is only beginning to be explored. At photovoltaic (PV) power plants, the forecast of solar irradiance is crucial for planning the power generation scheduling. In order to do this, we seek to forecast solar irradiance that take use of machine learning models and transformers. This thesis investigates the use of transformer models for solar irradiance forecasting, with the aim of optimizing energy in the digital age.

Significant barriers to the use of deep learning are removed by Pytorch Forecasting. Despite the fact that deep learning has won out in the fields of language processing, time series forecasting, and image processing. virtually invariably, GPUs are required for training neural networks, however they are not always readily accessible. Specifications for hardware are usually a significant obstacle too. But this problem may be solved by moving processing to the cloud, like Colab, where this experiment was carried out.

1.1 Transformer Model

To find context and meaning in sequential data, a neural network known as a transformer model records connections. It is driving a wave of machine learning advancements known as transformer AI, which is being used to translate text and speech almost instantly, make meetings and classrooms accessible to people with hearing loss and from different backgrounds, and help researchers understand the connections between genes and amino acids in proteins and DNA. They are taking the place of convolutional and recurrent neural networks (CNNs and RNNs), which were the most used deep learning model types five years ago. A robust neural network architecture called Transformers employs positional encoders to tag data pieces as they enter and exit the network. The attention units create an algebraic map of how each element is related to these tags. The word "self-attention" was almost adopted by Google researchers to describe their 2017 model since it is an effective tool for learning associations. The Transformer model was subsequently published by Google in 2017, Vaswani et al., (2017) . Transformers represent a significant departure from RNNs and CNNs, the two most widely used models for pattern recognition. Machine learning underwent a paradigm shift when the Google team trained their model on eight NVIDIA GPUs in just 3.5 days, spending a fraction of the time and money required to train prior models. Beyond comparable work published by a Facebook team using CNNs, it was a pivotal moment. Another Google team tried using a transformer to handle text sequences in both the forward and backward directions a year later. This effort established 11 new records and was incorporated into the Google search algorithm. International researchers were adapting BERT for use cases across many languages and industries.

Transformer models have proven to be valuable in various domains and tasks, including:

(a) Natural Language Processing (NLP): This encompasses a wide range of activities such as text categorization, named entity identification, question-answering, language modelling, summarization, translation, multiple-choice tasks, and text generation.

(b) Computer Vision (CV): Transformer models have also demonstrated effectiveness in tasks related to computer vision, including image segmentation, object identification, and image categorization.

(c) Audio Processing: Transformer models can be applied to audio-related tasks such as speech recognition software and voice classification.

(d) Multimodal Applications: Transformers are also suitable for multimodal tasks, such as Optical Character Recognition (OCR), document information extraction, table question answering, video classification, and visual question answering.

Transformer models are compatible with popular frameworks like JAX, TensorFlow, and PyTorch. This compatibility allows for seamless integration and transfer of models between frameworks. It is possible to train a model in one framework with just a few lines of code and then load it in another framework for inference. Additionally, models can be exported to file formats like ONNX and TorchScript, enabling their deployment in real-world applications.

Similar to how a Transformer is learned for machine translation, the model is trained via "teacher-forcing." This indicates that one prepends the final value of the past values to the future values as input to the decoder during training, moving them one place to the right. The model must forecast the subsequent target at each time step. Since there is no concept of decoder_start_token_id (we simply use the most recent value of the context as initial input for the decoder), the setup of training is similar to that of a GPT model for language.

We feed the decoder the final value of the past values at the moment of inference. The next step is to sample data from the model to produce a forecast for the following time step, which is then given to the decoder to make the subsequent prediction (also known as autoregressive generation).

1.2 Driving the Digital and Sustainable Transformation of the Energy System in Europe: Challenges and Opportunities

The European Green Deal and REPowerEU initiatives aim to transform the energy system in Europe towards sustainability and digitalization. This transformation requires leveraging digital technologies such as IoT devices, advanced connectivity, and cloud-edge computing. However, further efforts are needed to fully utilize the potential of these technologies while protecting privacy and data. Promoting connectivity and data exchange, enhancing cybersecurity and governance, and addressing energy consumption are crucial aspects of this transformation. Initiatives like the proposed Data Act and Data Governance Act play a vital role in ensuring a successful digital and sustainable energy transition in Europe EC, COM(2022).

To ensure the success of the digitalization of the energy system, attention must be given to cybersecurity, energy consumption, effective governance, digital rights and EU data sovereignty. Robust cybersecurity measures are essential to safeguard critical infrastructure and prevent unauthorized access. Addressing energy consumption concerns is vital to optimize energy efficiency and reduce wastage. Furthermore, designing effective governance frameworks ensures transparency, accountability, trust and compliance with data privacy regulations.

To realize a world where AI respects and preserves rights, the author proposed the concept of using AI for Socio-Economic opportunities and enhancing quality of life (#AI4SQL) as a volunteer expert involved in the creation of the Nigerian National AI Policy document, this Rights preserving AI concept was incorporated into the policy between 2022-2023 (NAIP 2022).

Our research aims to contribute to the digital and sustainable transformation of the energy system by developing an optimized model for solar irradiance forecasting which is aligning with the goals of the European Green Deal and REPowerEU initiatives. Our study offers valuable insights and practical solutions for achieving a clean and affordable energy future in Europe and the world.

1.3 Motivation.

The rising World population demands increased basic needs, including energy. According to United Nations Population projections, Nigeria will be the third most populated country by 2100 and would have the same population as the United States by 2050.This implies that electricity consumption in Africa will rise, both for humans and for energy-hungry robots. This current study is in furtherance to the paper on reviewing of AI and Blockchain applications in the energy industry (Akanni et al., 2023). We explored which AI model is the most accurate and effective at estimating how much sun power per unit area, measured in watts per square metre (W/m2) in SI units, will be available to be converted into electricity. We observed how the Time series Transformer model is effectively employed in NLP and machine translation, therefore we want to apply it for forecasting solar irradiance as well which from the best of our knowledge, this is the first time is done. Hence by combining AI models and renewable energy, we improve sustainability, optimize energy utilization in the digital age by using Transformer for solar irradiance forecasting in order to achieve a carbon-neutral economy. Hence optimization technique is focusing mainly on increasing a reliable and economic integration of PV by grid operators through accurate forecast of solar irradiance.

1.4 Statement of problem.

The problem addressed in this research is the need for accurate estimation of energy yield in photovoltaic (PV) systems to determine their viability as an alternative to traditional energy sources. Existing mathematical models for energy yield estimation are complex and require parameters that are difficult to obtain. Instead, the output of a PV system is influenced by meteorological data such as ambient temperature and solar irradiation, which can be challenging and expensive to measure. This necessitates the development of alternative prediction methods to accurately forecast solar irradiance. Forecasting solar power is a challenging task due to the variability of solar irradiation influenced by location, weather, and other meteorological factors. Accurate predictions of solar irradiance are crucial for the successful integration of solar energy with conventional generating sources. Energy forecasting, including solar power forecasting, is essential for effective grid management and power trading. Various statistical methods and theoretical models have been used for solar power forecasting, but the transformer model with the teacher forcing concept offers a unique approach that captures the context and relationships in solar and meteorological data. By optimizing the hyper-parameters of the Time series transformer, the model aims to achieve high forecasting accuracy, low MAPE and high R2 values for accurate solar power predictions.

1.5 Research Questions

- What are the main AI Models used in Solar forecasting? Which one is effective and superior?
- Do Time Series Transformer model for solar irradiance forecasting work?
- How can it compare with state of art paper using MAPE?
- What are some proposed recommendations to optimizing energy in the digital age?

1.6 Aims and Objective of the study

This research aims to address the challenge of accurately forecasting solar irradiance by introducing the teacher forcing concept to a time series Transformer model with an attention mechanism. The goal is to optimize energy generation by improving the accuracy of solar irradiance forecasts and comparing the performance with other machine learning models.

The objectives are:

- To develop a precise solar forecasting model that outperforms existing machine learning models.

- Compare our best model with other state-of-the-art MAPE forecasting results

The following Sustainable Development Goals (SDGs) are addressed in this work. The SDGs were established by the UN General Assembly to encourage cooperation among all nations and stakeholders.

- Goal 7.1: Assure that all people have access to modern, affordable energy services by the year 2030.

- Goal 7.2: Increase the proportion of renewable energy in the world's energy mix significantly by the year 2030.
- Goal 13: Climatic actions to limit and adapt to climate change.

1.7 Significance of the Study and contribution

The significance of this research study lies in addressing the limitations of traditional statistical techniques, neural network approaches, and theoretical models in forecasting solar irradiation. By leveraging the potential of deep neural networks, specifically the Time series Transformer model, this study aims to provide a practical and accurate solution for energy forecasting and optimization.

The implementation of the Time series Transformer model allows for forecasting of solar photovoltaic power by identifying connections and relationships within the data. The study utilizes performance evaluation metrics such as MSE, MAE, MAPE and R2 to assess the effectiveness and quality of the model.

Precise forecasts of solar irradiance are crucial for the effective integration of solar energy into the power grid. By utilizing the NASA Dataset and proposing the Time series Transformer technique, this research contributes to

- We developed a general Time series Transformer-based model for accurate solar irradiance forecasting models.
- We showed that our approach which is a unique combination of the Time series Transformer model with the teacher forcing idea and data pre-processing, providing an efficient and accurate solar predictor.
- The accuracy, performance and reliability of the model were investigated on the basis of standard performance evaluation metrics
- This study fills a gap in the literature by conducting a comprehensive evaluation of solar irradiance forecast models using the Time series Transformer and the teacher forcing idea.
- We showed that our time series Transformer-based model achieves state-of-the-art forecasting results.

According to our findings, Time Series Transformer model in test results shows effectiveness and superiority in explaining observed data, high forecasting accuracy with low mean absolute percentage error and high R2.
Overall, this research study provides valuable insights and practical solutions for forecasting solar irradiance, addressing the limitations of existing approaches, and advancing the field of energy optimization.

1.8 Limitation

After a thorough investigation of the response times of the current models, Transformer models require more resources to implement than more traditional ML model techniques. Especially for time series forecasting. No high-level API is available that interfaces with well-known frameworks like Google's Tensorflow or Facebook's PyTorch. For traditional ML, there is the Scikit-learn ecosystem, which provides a uniform user interface for professionals.

1.9 Organization

This thesis is made up of five chapters, as well as a conclusion, appendixes, and references.
Chapter 1: An outline of the study and its setting, research techniques, the research's objectives are given.
Chapter 2: The problem is addressed theoretically, and the chapter also offers review of related academic writing on the thesis's core subject. References to relevant sources are compared.
Chapter 3: This chapter provides an overview of the suggested solution and the Transformer Architecture, as well as a description of the study's methodology and a brief discussion of research methods.
Chapter 4: The training plan and performance are covered in this chapter.
Chapter 5: The model's effectiveness is assessed, and the outcome analysis is presented.
Conclusion and Recommendation of the report.

CHAPTER II

Related Research

2.0 Related Works

Solar irradiance predictions have been the subject of extensive investigation. Statistical, physical, and hybrid models are examples of traditional solar irradiance forecasting models. Short-term forecasting has traditionally relied on statistical models like Autoregressive Integrated Moving Average (ARIMA). However, there are limitations in the ability of these models to capture nonlinear relationships and are sensitive to outliers. Physical models, which use physical principles to model the behaviour of the atmosphere, have been shown to be accurate for long-term forecasting. However, physical models are complex and require extensive data inputs. Hybrid models, which combine statistical and physical models, have shown promising results.

Solar irradiance forecasting has recently used machine learning techniques. Short-term forecasting has been successfully accomplished using Support Vector Regression (SVR), Random Forest (RF), and Artificial Neural Networks (ANN). However, these models are limited by their inability to capture temporal dependencies.

Transformer models, introduced in NLP, have shown promising results in time series forecasting, such as teaching robots to translate words into French. They have been effectively applied to various time series forecasting applications, such as stock prices, electricity consumption, and wind generation. However, their application to solar irradiance forecasting has not been extensively studied. Many changes to Wen et al's Transformer model (Wen et al., 2022) have been successfully used to time series forecasting applications (Zhou et al., 2021; Li et al., 2019). Transformer models have demonstrated outstanding performance in capturing temporal dependencies.

This work uses a Multi-head Attention layer to understand temporal context information, in contrast to other studies (Brahma and Wadhvani, 2020; Alzahrani et al., 2017; Alharbi et al., 2021). Premalatha et al.'s study (Premalatha et al., 2016) shows a traditional ANN model with fully linked layers, which, in contrast to the attention matrix method, is unable to contextualize information in lengthy time series. The possibility of splitting the learning process into sunny and overcast days is shown by several studies (Zafare et al., 2021; Zafare et al., 2021; Wang et al., 2012). Furthermore, (Husein et al., 2019; Mendonça et al., 2020) explore the relationships between weather variables and solar irradiance and underline the advantages of employing meteorological data as input.

Since solar irradiance directly affects the amount of electricity generated by solar panels (Sharma et al., 2010), it is essential for a PV power plant to predict the level of solar irradiance ahead of time in order to optimize operational costs through generation scheduling (Liang et al., 2007).

Solar Forecasting is a technique for foreseeing the solar irradiation components for a certain PV installation. The three basic approaches are statistical time series, physical approaches, and ensemble approach. We focus on the statistical time series approach.

Using a statistical time series approach, while retaining long-term dependencies, statistical time series methods have limits in their capacity to precisely connect time series input and output for both long-term and short-term periods. Time series Transformer address these limitations in machine learning and deep learning techniques by handling long term dependencies well.

Due to significant computing requirements, using complicated physics-based models is often seen as costly (Prema et al., 2015). As a result, the goal of this research is to develop models that take meteorological information into account and are capable of accurately forecasting solar irradiance using low-cost machine learning approaches.

2.1 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are composed of multiple layers, each containing a specific number of neurons. Figure 2.1 illustrates a simple neural network, where each color represents a layer and each circle represents a neuron. The first layer of an ANN is the input layer, which receives a vector of input features. An important advantage of neural networks is that input data does not require preprocessing before being fed into the network. The data then propagates through hidden layers, ultimately reaching the final output layer. The number of neurons in the output layer depends on the problem at hand. ANNs can be utilized for both classification and regression tasks, based on the chosen loss function.

The propagation of data through a neural network is determined by the network parameters, namely the weights (W = {w_1, w_2, ..., w_i, ..., w_{m-1}}) and biases (B = {b_1, b_2, ..., bi, ... , b_{m-1}}), where m represents the number of layers in the network. Each weight (wi) is a matrix with dimensions $l \times k$, and each bias (b_i) is a vector with dimension l, where k is the number of neurons in the previous layer ($_{i-1}$) and l is the number of neurons in layer i.

The value of each neuron in layer i is calculated as a linear combination of the neurons from the previous layer, followed by a non-linear activation function. Common activation functions include the hyperbolic tangent, rectifier (ReLU), and sigmoid functions. The weights and biases are the parameters optimized during the training process. Equation 2.1 depicts this relationship,

$$a_i = f\,(w_{i-1}\,a_{i-1} + b_{i-1}) \qquad (2.1)$$

where a_i represents the activation vector representing the neuron values in layer i, f is the activation function, and w_{i-1} and b_{i-1} are the corresponding weights and biases, respectively.

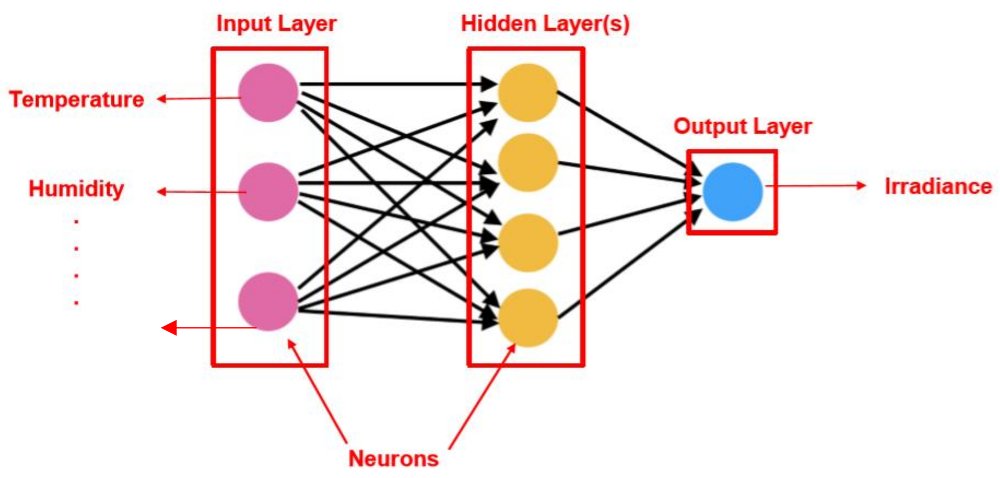


Figure 2.1: A simple neural network.

A crucial characteristic of neural networks is their ability to approximate functions. The Universal Approximation Theorem states that any function f can be approximated by a neural network with a sufficient number of neurons and layers. (Milind et al., 2020)

2.2 Optimization Problem

The problem of solar irradiance forecasting can be formulated as an unconstrained optimization problem. The objective function in Equation 2.2 is the mean squared error (MSE), where the goal is to minimize the average squared difference between the actual and forecasted values of irradiance. The predicted value is a continuous output from the neural network's output layer. Since the layers of a neural network are interconnected through Equation 2.1, the error propagates backward through the layers. Consequently, the parameters of the ANN, specifically the weights and biases, are adjusted using a backward-propagation mechanism known as backpropagation. [55]

$$\min_{W,B} \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.2}$$

In the equation, ˆy and y represent the forecasted and actual values of solar irradiance in W/m2, respectively. n is the number of samples, and W and B represent the sets of weights and biases, respectively.

2.3 Transformer

The Transformer architecture addressed the issue of preserving long-term dependencies by leveraging (a). self-attention mechanisms to retain word-to-word relation and (b). positional encodings to represent each word’s position. This enables parallel computation over the entire text without disrupting the order. The Transformer has an encoder for input text and a decoder for generating text. (Vaswani et al., 2017)

2.4 Time Series Forecasting

Recent years have seen a breakthrough in time series forecasting research using various deep learning algorithm modifications. Numerous practical fields, including weather, economics, agriculture, transportation, and even exact scientific reasons, have embraced applications.

All of the natural language processing models were replaced with a transformer. Given the similarity between completing of text andty6 forecasting of time series data, strategy based on attention was also used in Time Series. (Wu et al., 2021) applied a Transformer-based approach to forecast influenza cases, demonstrating its superiority compared to other sequential models. (Grigsby et al., 2021) developed Transformer-based models for time series forecasting, considering distinct spatial relationships between variables and achieving improved forecasting results. In their study, Haoyi (Zhou et al., 2021) developed a transformer model called Informer specifically designed for predicting long sequence time-series data. This model was applied to tasks such as electricity consumption planning, which requires a high prediction capacity. The term "prediction capacity" refers to the model's ability to accurately capture long-range dependencies between the output and input variables efficiently.

2.5 PyTorch Forecasting for Time Series Forecasting

Though deep learning has surpassed conventional approaches in time series forecasting tasks, deep learning architectures have not yet become the norm for time series forecasting tasks, despite dominating computer vision and language processing workloads. The lack of a high-level API that would operate with well-known frameworks like PyTorch or Tensorflow has been a key barrier, in addition to the hardware requirements, making it rather challenging to leverage neural networks over the conventional approaches (easy to use in the scikit learn ecosystem). By giving PyTorch a high level API that can easily use the panda’s data frame, PyTorch Forecasting finds a solution to the issue. The package's foundations in PyTorch Lightning and PyTorch APIs make learning it simpler. Modern time series are made easier with Pytorch Forecasting. using neural networks for predicting in both academic and real-world scenarios. The package has some intriguing clauses, such as: a class for time series datasets that abstracts away the processing of variable transformations, missing values, random subsampling, different history lengths, etc.

Therefore, in order to train your model in PyTorch, no specialized expertise of dataset creation is needed. Basic training of time series models is provided through a base model class, along with logging in Tensor board and general visualizations like actual vs. predicted values and dependency charts. There are numerous neural network topologies for time series forecasting that have been improved for real-world application and come with built-in interpretation capabilities, time series metrics with multiple horizons for scalability, the networks are made to function with PyTorch

Lightning, which out-of-the-box supports training on CPUs as well as single and multiple (distributed) GPUs. (Kasper, 2022)

2.6 Transformer model for Solar irradiance prediction

To utilize a transformer model for predicting solar irradiance based on a dataset of solar radiation and weather, the following steps need to be undertaken:

Data pre-processing: The dataset should be cleaned and, if necessary, normalized or standardized as part of the pre-processing stage. Additionally, the data might need to be transformed into a format compatible with the transformer model. This involves tasks such as including the timestamp, standardizing the solar radiation dataset, and normalizing the solar irradiance data.

Dataset separation: Split the dataset into training and test sets. The training set is used to train the transformer model, while the test set is used to evaluate its performance.

Define the transformer-based model: Specify the characteristics of the transformer model, such as the number of layers, attention heads, and hidden layer dimensions. Additionally, define the input and output layers of the model.

Model training: Train the transformer model using the training set. Specify the optimizer, loss function, and any other relevant training hyper parameters.

Model evaluation: Assess the performance of the trained model using the test set. Calculate performance metrics like mean squared error (MSE) or mean absolute error (MAE) to measure the accuracy of the model.

Prediction application: Once the model has been trained and evaluated, it can be applied to make predictions on new data. However, this fresh data needs to undergo the same pre-processing steps as the training data before the model can generate accurate predictions (Vaswani et al., 2017)

The Time Series Transformer model is a probabilistic vanilla encoder-decoder Transformer for time series forecasting. It adds a distribution head on top of the former, which can be used for time-series forecasting. Note that this is a so-called probabilistic forecasting model, not a point forecasting model. This means that the model learns a distribution, from which one can sample. The model doesn’t directly output values. (Niels and Kashif, 2022)

CHAPTER III

3.0 Methodology/Materials

3.1 From Machine learning - Neural Networks to Deep Learning-Transformers

Artificial Neural Networks are brain-inspired systems which are intended to replicate the way we humans learn.

Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use.

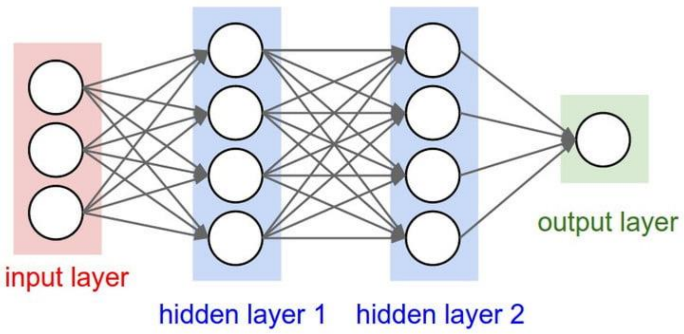


Figure 3.1: Shows the basic structure of a Neural Network

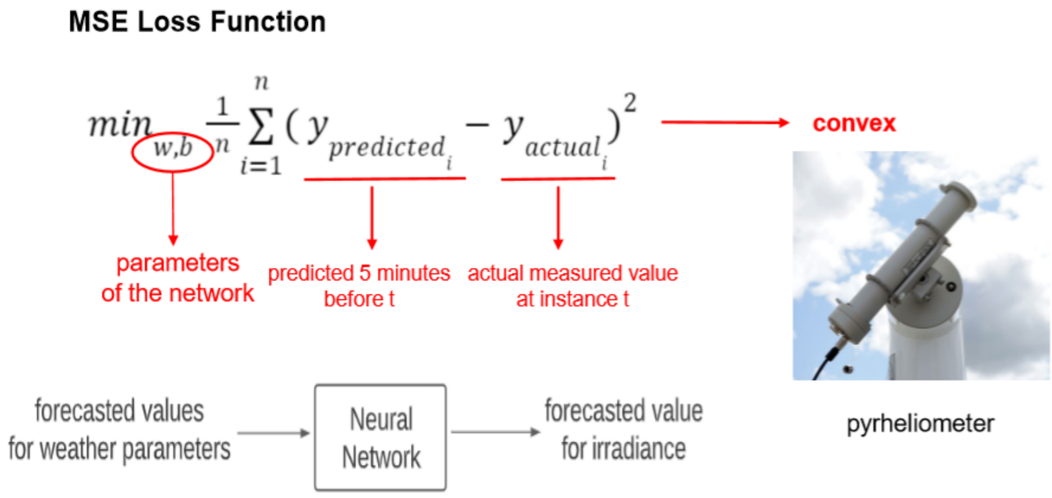


Figure 3.2: Shows the MSE loss function of ML models used to forecast irradiance value

3.2 Transformer Architecture

In deep learning, a transformer is a model. It stands out for adopting self-attention and differently valuing the importance of each component of the input data (which includes the recursive output). Natural language processing is where it is most frequently employed.

Transformer models are a sort of neural network that examine connections in sequential data in order to understand context and meaning.

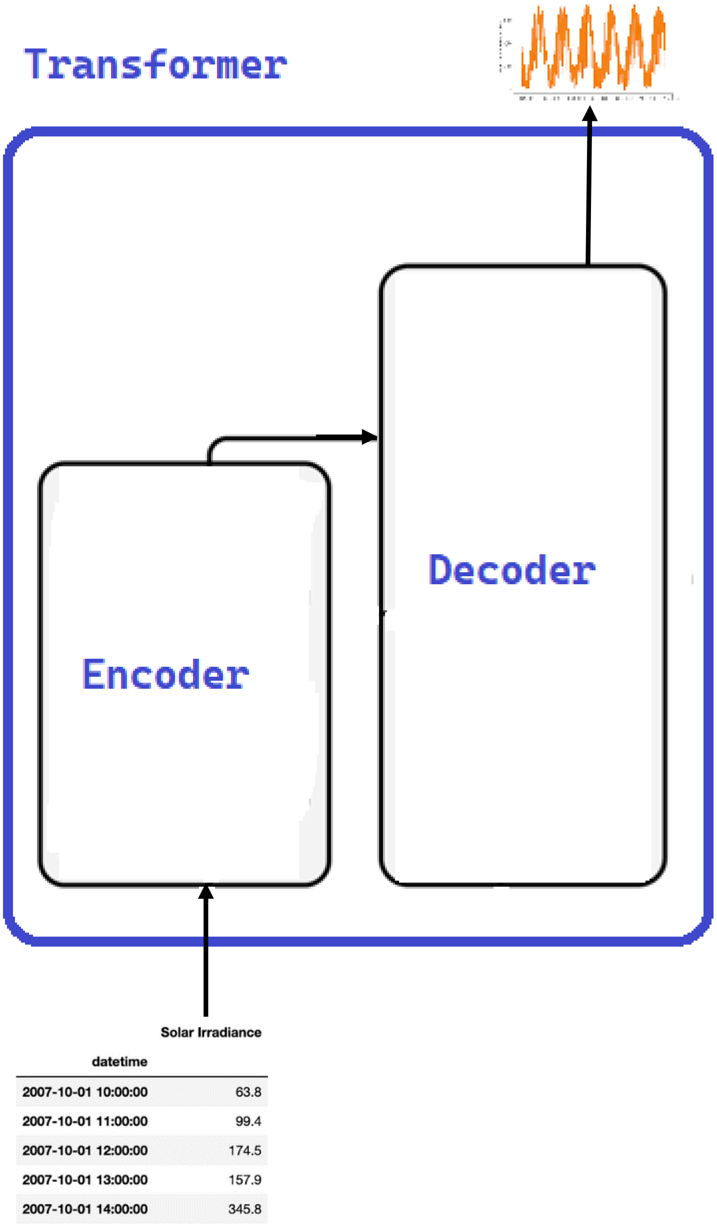


Figure 3.3: Shows the basic structure of a time series transformer

3.2.1 Encoder/decoder architecture

The encoder component of the model consists of several parts, including positional encoding, an input layer, and N identical encoder layers. The input layer converts the input time series data into a vector using a fully connected network, which is necessary for the multi-head attention mechanism to operate. Each encoder layer also includes a fully connected feed-forward neural network after the multi-head attention layer. The output of each encoder layer is passed as input to the next layer, and all layer outputs have the same dimensions. The decoder component, on the other hand, consists of an input layer, N identical decoder layers, and an output layer. Each decoder layer includes two multi-head attention layers and a feed-forward neural network. The initial attention layer of each decoder layer receives input from the output of the preceding layer, while the second attention layer uses the output from the encoder stack as its input.

In the encoder component of the model, the input variables such as solar radiation, date, and time are transformed into a vector representation. The encoder stack aims to capture the relationships between these elements, enabling their conversion from historical input data to the latent space, which serves as the input for the subsequent part of the model.

By combining the date, time of year, historical data, and the vector representation from the encoder, the decoder stack acts as a time machine, forecasting solar irradiance.

3.3 Positional encoder

Transformers employ positional encoders to identify data elements entering and leaving the network. Attention units then follow these tags and create an algebraic map showing how one element connects to the others.

A vector form known as a positional encoding contains information about the relative positions of letters within a target sequence. It's characterized as a function of kind $f: \mathbb{R} \rightarrow \mathbb{R}; d \epsilon \mathbb{Z}, d > 0$, where d is an even positive number.

$$(f(t)_{2k}, f(t)_{2k+1}) = (\sin(\theta), \cos(\theta)) \qquad \forall \kappa \in \left\{0,1, \ldots\ldots, \frac{d}{2} - 1\right\}$$

(1)

$$where\ \theta = \frac{t}{r^k}, r = N^{\frac{2}{d}}.$$

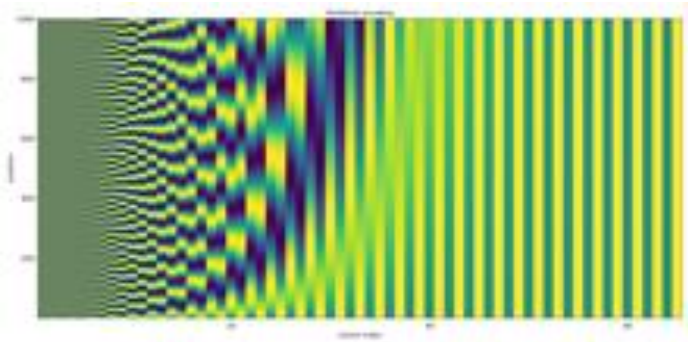


Figure 3.4 illustrates a sine wave positional encoding scheme with parameters N=10000 and d=100. Image source: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

The dimension the vector encoding a time series data, is denoted by the free parameter N, which should be much larger than the maximum k.

The most important fact is that every encoded location may be utilized by the transformer to compute the linear sum of its neighbours, which can then be used to provide attention weights for the attention mechanism.

Positional encoding can be simply put together and expressed as

$$PE_{(x,2k)} = \sin\left(\frac{x}{N^{2k/d}}\right)$$

$$PE_{(x,2k+1)} = \cos\left(\frac{x}{N^{2k/d}}\right)$$

Hence, $PE_{(x,2k)} = \sin(\lambda_z . t)$

$$PE_{(x,2k+1)} = \cos(\lambda_z . t) \tag{2}$$

Where $\lambda_t = \frac{1}{N^{2k/d}}$

3.3.1 Scaled dot-product attention

Scaled dot-product attention is a component of the transformer model that involves learning three weight matrices: query weights, key weights, and value weights. These matrices are used to create query vectors, key vectors, and value vectors for each token by multiplying them with the input time series data. The resulting values are then normalized using softmax and divided by the square root of the dimension of the key vectors to obtain attention weights. The output of the attention mechanism is a weighted sum of the input values, where the weight is determined by combining the query input and the corresponding key input.

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

3.3.2 Multi-head attention

The multi-head attention technique is employed to project the query, key, and value vectors into multiple linear spaces. This technique performs self-attention in parallel multiple times, with each head utilizing unique learned matrices for query, key, and value to capture complex relationships. The outputs of these heads are concatenated together. To enhance the model's accuracy, multiple attention matrices are combined into a single output as described in equation (4). The position-wise fully connected feed-forward network block, represented by equation (5), operates on the concatenated outputs.

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^o$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{4}$$

with use made of the projection matrices

$$W_i^j \in \mathbb{R}^{d\times d_j}; j \in \{Q,K,V\}$$

And the weight matrix

$$W^o \in \mathbb{R}^{hd_v\times d}$$

This time, the input value to the attention block is represented by the matrix V, the input key is represented by the matrix K, and the input query is represented by the matrix Q.

The value d_k represents the size of the key input. The matrices $W_{i,}^Q$, $W_{i,}^K$, $W_{i,}^V, W_{i,}^V, and\ W_{i,}^Q$

indicate the model parameters that have been learnt for the projection of the features.

$$FFN(x) = GeLU\ (xW_1 + b_1)W_2 + b_2 \tag{5}$$

While the GeLU (Gaussian Error Linear Unit) in the aforementioned equation denotes nonlinearity in the model, the matrix W and b represents weights and biases.

3.4 Loss Function:

A loss function may be used to determine the degree to which an algorithm accurately reflects a dataset. Pseudo-Huber, Huber Loss, Mean Squared Logarithmic Error Loss, L1 Loss, L2 Loss, Mean Absolute Error Loss, and the complete positional encoding function are some of the numerous types of Regression loss functions. L1 and L2 are two common loss functions used in deep learning and machine learning to lower error. The Least Absolute Deviations (LAD) or cost is defined as the Mean of these Absolute Errors (MAE).

The cost is measured by the Mean of these Square Errors (MSE), and L2 is referred to as Least Square Errors (LS). The disadvantage of the L2 norm is that in case of outliers, these points will be mostly accountable for the primary component of the loss.

$$L1LossFunction = \sum_{i=1}^{n} |y_{true} - y_{predicted}| \tag{6}$$

$$L2LossFunction = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2 \tag{7}$$

3.5 Solar Irradiance Transformer Model

The Transformer model has gained widespread use in neural networks for tasks like natural language translation and has recently been applied to time series data, such as solar irradiance. Its attention mechanism plays a crucial role in connecting relevant features within sequential input. The model is composed of two fundamental building blocks: positionally fully connected feed-forward network blocks and self-attention blocks.

The self-attention blocks consist of four layers: a normalization layer, a multi-head attention layer, a dropout layer, and a residual connection. Similarly, the positionally fully connected feed-forward network blocks include a normalization layer, a residual connection, a dropout layer, and two fully connected (dense) layers. These building blocks are stacked together to create the encoder and decoder, which are the core components of the model.

The self-attention block establishes connections between the query output and the key-value output, allowing the combination of query, key, and value inputs into a unified output.

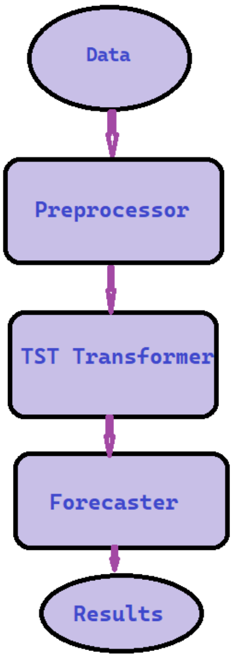


Figure 3.5: Flow diagram of TST Transformer for Solar forecasting process

3.6 Metrics

Different metrics are used to assess the precision of predicted solar irradiance, which include mean squared error, mean absolute error, and R squared. The mean absolute error specifically measures the accuracy of solar irradiance in W/m^2. When computing these metrics, previous time steps are not taken into account. However, during training, all time steps are utilized to calculate gradients and provide crucial data to the model.

The performance of the implemented model on the test dataset was evaluated using commonly used metrics in time-series models, namely the MAE loss function, RMSE,

MSE, and R-squared. These metrics provide valuable insights into the model's ability to generate accurate predictions on unseen data.

The MAE loss function measures the average absolute difference between the predicted values (ŷi) and the actual values (yi), as represented by equation (8).

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

A lower MAE signifies a more accurate model, indicating a smaller average error between the predicted and actual values. MAE calculates the average absolute difference between the predicted values and the actual values. It measures the average magnitude of the errors without considering their direction. RMSE is similar to MAE, but it takes the square root of the average of the squared differences between the predicted values and the actual values. It penalizes larger errors more heavily than MAE and provides a measure of the standard deviation of the residuals.

Formulas (9) and (10) introduce two coefficients used to assess the accuracy of prediction models

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{N} (y_i - \widehat{y_i})^2 \tag{9}$$

N represents the total number of observed samples, yi is the actual value, and ŷi is the Predicted value. The goal of these metrics is to minimize the sum of squared errors (MSE), which quantifies the deviation between the actual and predicted values, as shown in equation (5)

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \widehat{y_i})^2}{N}} \tag{10}$$

.

The RMSE is the square root of MSE and provides a measure of error in the same units as the baseline values from the dataset. RMSE yields lower absolute values and is computationally efficient compared to MSE.

In addition to MAE, RMSE, and MSE, R-squared (R^2) is another commonly used metric in time-series models. It measures how well the model fits the data by indicating the proportion of variance in the dependent variable that can be explained by the independent variables. R-squared is calculated using the formula:

$$R^{2} = 1 - (SS\_res / SS\_tot) \quad (11)$$

where SS_res represents the sum of squares of residuals and SS_tot represents the total sum of squares. Where SS_res is the sum of squares/N ie MSE and SS_tot is Sum of Squared Error from mean/N. A higher R-squared value (closer to 1) indicates a better fit of the model to the data, while a lower value suggests that the model explains less of the variance in the data.

Including R-squared in the evaluation of time-series models provides an additional measure to assess the goodness-of-fit and predictive capability of the model. R-squared measures the proportion of the variance in the dependent variable that can be explained by the model. It ranges from 0 to 1, where 1 indicates a perfect fit and 0 indicates no relationship. It complements the MAE, RMSE, and MSE by providing an evaluation of the proportion of variability in the data that can be attributed to the model's predictions. Hence forecast accuracy of models were compare with MAE, RMSE, and MSE first then reliability metric R2.

The above performance metrics were used first to derive the best model for solar irradiance

forecasting then the MAPE of best model which is transformer in this study was compared with different other models authors used for solar energy forecasting.

Mean Absolute Percentage Error (MAPE) measures the average percentage difference between

the predicted values and the actual values. It is useful when you want to understand the relative magnitude of the errors compared to the actual values.

3.7 Methodology

The following procedural processes are part of the methodology used in this study.

The Kaggle NASA Solar Radiation prediction dataset, available in CSV format, consists of columns containing solar radiation and weather data such as temperature, humidity, pressure, wind speed, and wind direction.

The solar dataset underwent preprocessing to extract nine relevant features that will be utilized in our model for predicting solar irradiance. These features include temperature, pressure, humidity, wind direction in degrees, sunrise time, sunset time, and time-related data such as the length of daylight. Additionally, we extracted information such as hours and minutes for dawn and sunset timings, the hour of the day, the length of the day, and the week of the year from the available time data.

We then trained ten different machine learning model algorithms using the dataset. Specifically, for the Time Series Transformer model, we separately prepared and normalized the data as part of the required data preprocessing techniques before inputting it into the Transformer model.

Even before we down select to a specific Machine learning model, we prepared a prediction algorithm that takes in our data and makes a prediction. We used scikit-learn, to easily swap out different models and maintain the same higher-level structure to the program. We desire an algorithm that will predict values radiation for a given set of inputs.

In this investigation, we will try several ML models and compare their performance to evaluate the best algorithm to predict solar radiation. The specific Machine learning models used to exploit the NASA Dataset are Linear regression, Random Forest Regression, Neural Network Regression, Support Vector Regression, Gradient Boosting Regressor, K-Nearest Neighbors Regressor, Decision Tree Regressor, Ridge Regression, Lasso Regression, ElasticNet Regression.

To train the ML prediction algorithm, we implement a split train/test methodology to prevent bias in the learning. The dataset is split into a randomly sampled pool of data points. 80% of those points are used for training, the remaining 20% is used for validation of the training data. So the test data is not necessarily continuous time, but rather a random selection of points from the set.

For EDA purposes, we use the entire dataset (including training and test points) to visualize algorithm performance over time. This is inherently biased, since some of the points we will see will have been points that the algorithm has already trained on and potentially optimized to. However, we validate the algorithm accuracy against the subset of testing points (which the were not used for training), so we can still be confident in evaluating the performance using the accuracy metric and by keeping this potential bias in mind.

The results of the 10 machine learning model were recorded in the experiment excel sheet.

3.7.1 Sourced Datasets: CSV format (Kaggle NASA's Solar Radiation prediction dataset). The NASA datasets comprise solar and meteorological data collected over a four-month period (September through December 2016) as part of the NASA Space Apps Challenge hackathon for the NASA weather station. It may be freely found here [25]. NASA dataset is a single file that contains 32,686 rows and 11 columns of which 4 are in Decimal form ,4 are in Date Time format and 4 are Integer.

	UNIXTime	Data	Time	Radiation	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	TimeSunRise	TimeSunSet
0	1475229326	9/29/2016 12:00:00 AM	23:55:26	1.21	48	30.46	59	177.39	5.62	06:13:00	18:13:00
1	1475229023	9/29/2016 12:00:00 AM	23:50:23	1.21	48	30.46	58	176.78	3.37	06:13:00	18:13:00
2	1475228726	9/29/2016 12:00:00 AM	23:45:26	1.23	48	30.46	57	158.75	3.37	06:13:00	18:13:00
3	1475228421	9/29/2016 12:00:00 AM	23:40:21	1.21	48	30.46	60	137.71	3.37	06:13:00	18:13:00
4	1475228124	9/29/2016 12:00:00 AM	23:35:24	1.17	48	30.46	62	104.95	5.62	06:13:00	18:13:00

Figure 3.6a: Shows the first 5 rows of the NASA dataset

```
                           Radiation  Temperature  Pressure  Humidity  \
UNIXTime
2016-09-01 00:00:08-10:00       2.58         51.0     30.43       103
2016-09-01 00:05:10-10:00       2.83         51.0     30.43       103
2016-09-01 00:20:06-10:00       2.16         51.0     30.43       103
2016-09-01 00:25:05-10:00       2.21         51.0     30.43       103
2016-09-01 00:30:09-10:00       2.25         51.0     30.43       103

                           WindDirection  WindSpeed  Hour  DayLength
UNIXTime
2016-09-01 00:00:08-10:00          77.27      11.25     0    45060.0
2016-09-01 00:05:10-10:00         153.44       9.00     0    45060.0
2016-09-01 00:20:06-10:00         142.04       7.87     0    45060.0
2016-09-01 00:25:05-10:00         144.12      18.00     0    45060.0
2016-09-01 00:30:09-10:00          67.42      11.25     0    45060.0
```

Figure 3.6b: Shows the first 5 rows and 9 features of the NASA dataset used to train the 10 AI Models

	A	B	C	D
1	DateTime	Radiation	Normalized Radiation	
2	9/29/2016 23:55	1.21	6.25E-05	
3	9/29/2016 23:50	1.21	6.25E-05	
4	9/29/2016 23:45	1.23	7.50E-05	
5	9/29/2016 23:40	1.21	6.25E-05	
6	9/29/2016 23:35	1.17	3.75E-05	
7	9/29/2016 23:30	1.21	6.25E-05	
8	9/29/2016 23:25	1.2	5.62E-05	
9	9/29/2016 23:20	1.24	8.12E-05	
10	9/29/2016 23:15	1.23	7.50E-05	
11	9/29/2016 23:10	1.21	6.25E-05	
12	9/29/2016 23:05	1.23	7.50E-05	
13	9/29/2016 23:00	1.21	6.25E-05	
14	9/29/2016 22:55	1.22	6.87E-05	
15	9/29/2016 22:50	1.21	6.25E-05	
16	9/29/2016 22:45	1.23	7.50E-05	
17	9/29/2016 22:40	1.22	6.87E-05	

Figure 3.7a: Shows the first 17 rows of the NASA dataset used for Transformer model

	A	B	C	D
1	DateTime	Radiation	Normalized Radiatio	
2	9/29/2016 23:55	1.22	6.87E-05	
3	9/29/2016 23:50	1.21	6.25E-05	
4	9/29/2016 23:45	1.23	7.50E-05	
5	9/29/2016 23:40	1.22	6.87E-05	
6	9/29/2016 23:35	1.21	6.25E-05	
7	9/29/2016 23:30	1.22	6.87E-05	
8	9/29/2016 23:25	1.22	6.87E-05	
9	9/29/2016 23:20	1.2	5.62E-05	
10	9/29/2016 23:15	1.2	5.62E-05	
11	9/29/2016 23:10	1.2	5.62E-05	
12	9/29/2016 23:05	1.21	6.25E-05	
13	9/29/2016 23:00	1.22	6.87E-05	
14	9/29/2016 22:55	1.22	6.87E-05	
15	9/29/2016 22:50	1.22	6.87E-05	
16	9/29/2016 22:45	1.24	8.12E-05	
17	9/29/2016 22:40	1.23	7.50E-05	
18	9/29/2016 22:35	1.23	7.50E-05	
19	9/29/2016 22:30	1.23	7.50E-05	

Solar_SHIFTED

Figure 3.7b: Shows the first 19 rows of the 1 hour shifted dataset used for Transformer model

3.7.2 Data Cleaning (Manual): Extracted Date-Time column and the Radiation column, then normalised the radiation and code was used to carry out exponential smoothing of the data for Transformer model training after the training was done and results obtained the dataset was shifted for 1 hour and the transformer model was trained with it the result for the Validation loss of both the dataset and shifted dataset was plotted to see percentage change from sample to sample of the transformer model.

3.7.3 Notebook Procedures: The process described in preceding sections on using Transformer model to predict Solar radiation was carried out.

3.8 Data Pre-processing and Feature Engineering

The solar data was pre-processed to extract 9 features which will be used in our model to predict solar irradiance. Some features that will be used include temperature, pressure, humidity, wind direction in degrees, sunrise time, sunset time along with the time data while the length of day sunlight was extracted. In addition to the hours and minutes for dawn and sunset timings, we also retrieved the hour, day's length, and week of the year from the time shown in the data. As soon as the information was imported, we initially performed feature engineering, which involved converting time and date parameters into a more usable format and adding a few columns that would be beneficial for modelling, visualizing, and analyzing the data. We construct a matrix that determines the correlation between every pair of potential extracted feature in order to better comprehend the patterns and relationships in the data. To finally prepare the data to be used for model training, we cleaned it, checked for null value in the dataset, ingest data for exploratory data analysis and 10 machine learning model Algorithm was trained with the dataset. For Time series transformer, we prepare it separately and normalised it as part of the data pre-possessing techniques required, before data input into the Transformer model

For data visualization, plotting libraries are loaded. The influence of each measurement on the others is then determined using Pearson correlations and the visualization of each measurement. To remove pointless information and pinpoint the set's most important traits, a Pearson correlation matrix is first created.

3.9 Setup of the experiment

Our Time Series Transformer model processes a set of historical data into a set of forecasts for the future. Our analysis of real-world data demonstrates that our model exceeds cutting-edge techniques in terms of accuracy and effectiveness.
With the help of Pytorch, the Tensorflow, Keras API, and Pytorch-ignite (0.4.10), the Time series Transformer model was developed. For the GPU cloud environment, the author chose Colab. Additionally, used in the model development process are crucial Python modules including Numpy (1.18.5), Scikit-Learn (0.22.2), matplotlib, and Pandas (1.0.5).

To train the model, we have already pre-processed the dataset by normalizing the solar radiation column of the NASA dataset. We first define the transformer model, specifying layers' number, attention heads; number, hidden layers' size. Then compile the model, specifying the optimizer and loss function that will be used during training.

Next, the model is trained using the fit method, which takes the training data as input. The model is then evaluated on the test set using evaluate method. Finally, the model is used to make predictions on the test set using the predict method to predict the future solar radiation.

We define a Transformer-based time series model, that is developed to handle sequential data such as time series. The model, named TransAm, consists of several components:

1. Positional Encoding: This component is used to add position information to the input data, which is essential for the Transformer model to understand the order of the time steps.

2. nn.TransformerEncoderLayer: This component is a single layer of the transformer encoder, which applies self-attention to the input data and performs multi-head attention to the input data.

3. nn.TransformerEncoder: This component is a stack of transformer encoder layers that applies multi-head self-attention to the input data

4. nn.Linear: This component is a linear layer that is used as a decoder that takes the output from the TransformerEncoder and produces the final output.

The set up imports the necessary libraries: torch, torch.nn, numpy, pandas, time, math, pyplot, and some classes and functions from ignite.metrics module. Seeds are set for reproducibility of the results. Definitions of input_window, output_window, and batch_size are provided. These variables determine the size of the input window, output window, and batch size, respectively. The device is set to "cuda" if available, otherwise "cpu".

The Positional encoding class is defined, which adds positional encodings to the input data. It initializes a positional encoding matrix pe with dimensions (max_len, d_model), where max_len is the maximum length of the sequence and d_model is the feature size. The positional encodings are based on sine and cosine functions of different frequencies. The forward method adds the positional encodings to the input tensor.

The TransAm class is defined, representing the Transformer-based model for solar irradiance prediction. It inherits from nn.Module. The model consists of a positional encoding layer, a transformer encoder layer, and a linear decoder layer. The encoder layer is applied num_layers times. The forward method performs the forward pass of the model, applying the positional encoding, transformer encoding, and linear decoding to the input sequence. The init_weights method initializes the weights of the linear decoder layer.

The _generate_square_subsequent_mask method generates a mask matrix for the transformer encoder layer. It creates a square mask where each element below the main diagonal is set to -inf and each element on or above the diagonal is set to 0.

In summary, the code sets up the necessary modules and classes for the Transformer-based model for solar irradiance prediction. It defines the positional encoding layer, the Transformer encoder layer, and the linear decoder layer. The model takes an input sequence, applies positional encoding, transformer encoding, and linear decoding to generate the predicted output sequence.

the create_inout_sequences function generates input-output sequences for training the time series transformer model. The get_data function generates the solar irradiance data and splits it into training and testing sets. These functions are used to prepare the data for training and evaluating the model. We train a time series transformer model using the specified training and testing data. It tracks the training and testing losses during the training process and plots the loss curve at the end. We train the time series transformer model for the specified number of epochs (100 epochs), evaluates its performance on the validation data, and saves the best model based on the validation loss. Additionally, the code generates plots and predicts future values at regular intervals during training. We evaluate the trained transformer model on the test data, calculates the loss, generates a plot comparing the predicted and true values, and saves the predicted and true values as CSV files.

CHAPTER IV

Results and Discussions

4.0 EDA Results and Discussions

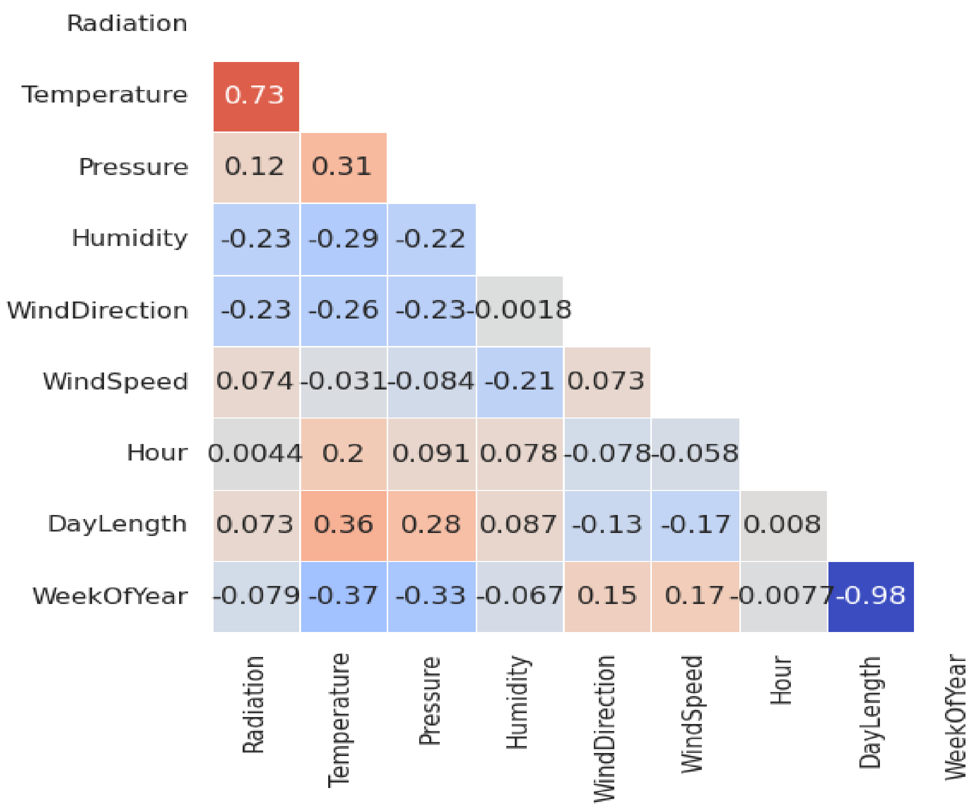


Figure 4.1: shows the Pairwise correlation matrix of the features used in NASA dataset

Correlation matrix shows that temperature is the highest positive value of 0.73 is relevant and Humidity and wind direction are lowest which mean when there high humidity and wind direction it suggests presence of cloud cover so when the sky is not ckear, there will be low solar radiation but as temperature increases the solar radiation increases.

The provided charts clearly indicate a strong correlation between temperature and solar irradiance. While the relationships between pressure and solar irradiance are not as clear, there appears to be a negative correlation between humidity and solar irradiance; temperature and pressure.

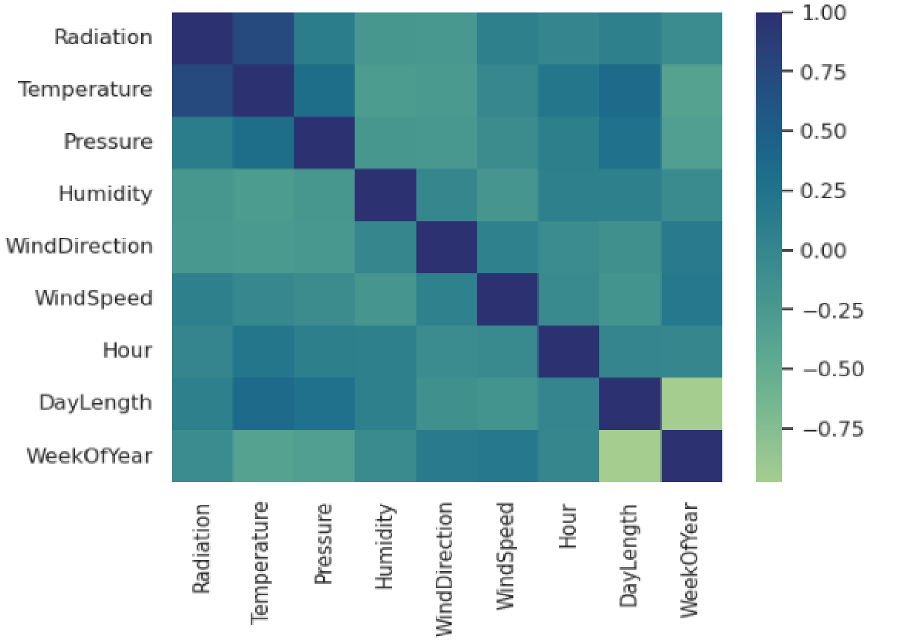


Figure 4.2: the graph showing the heat map correlation Matrix of the features used in NASA dataset

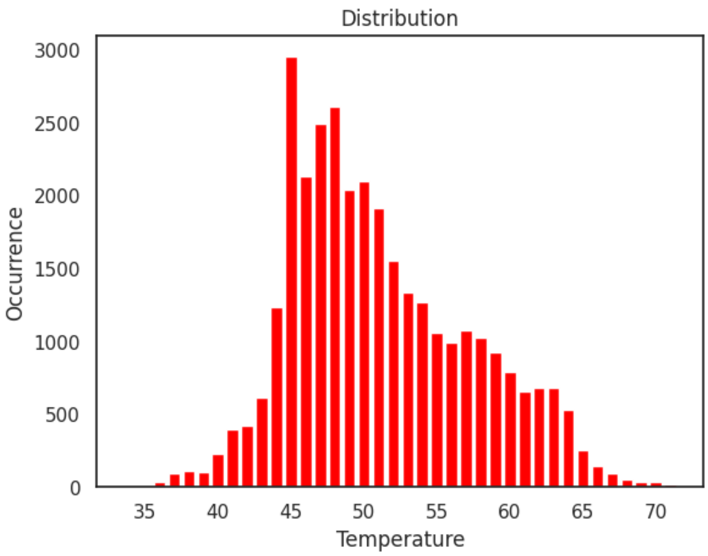


Figure 4.3: Distribution of Temperature and the number of occurrence in the NASA dataset

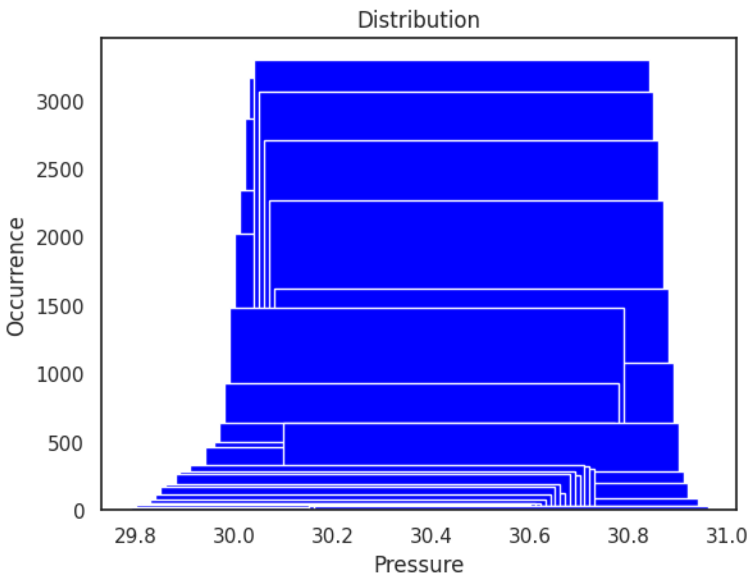


Figure 4.4: Distribution of Pressure and the number of occurrence in the NASA dataset

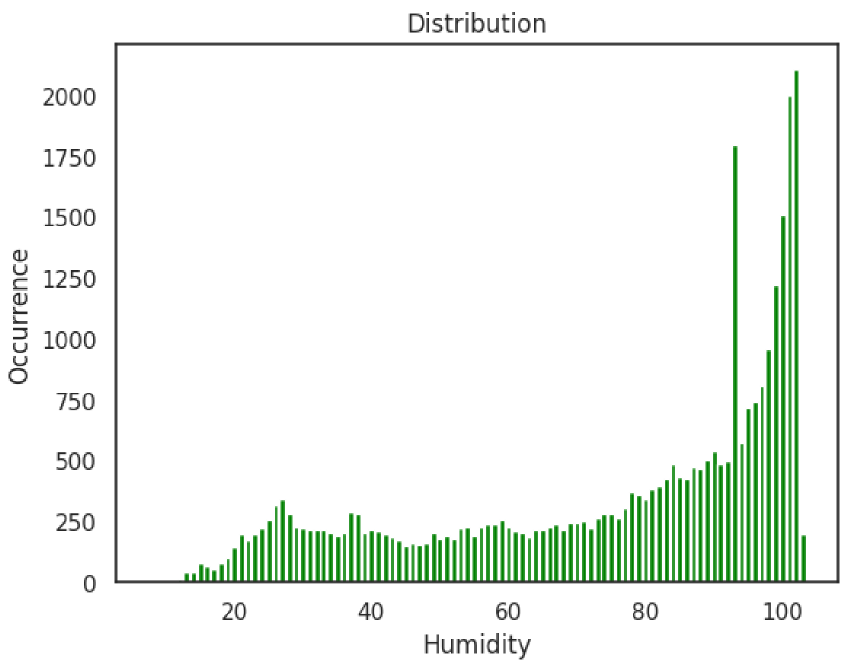


Figure 4.5: Distribution of Humidity and the number of occurrence in the NASA dataset

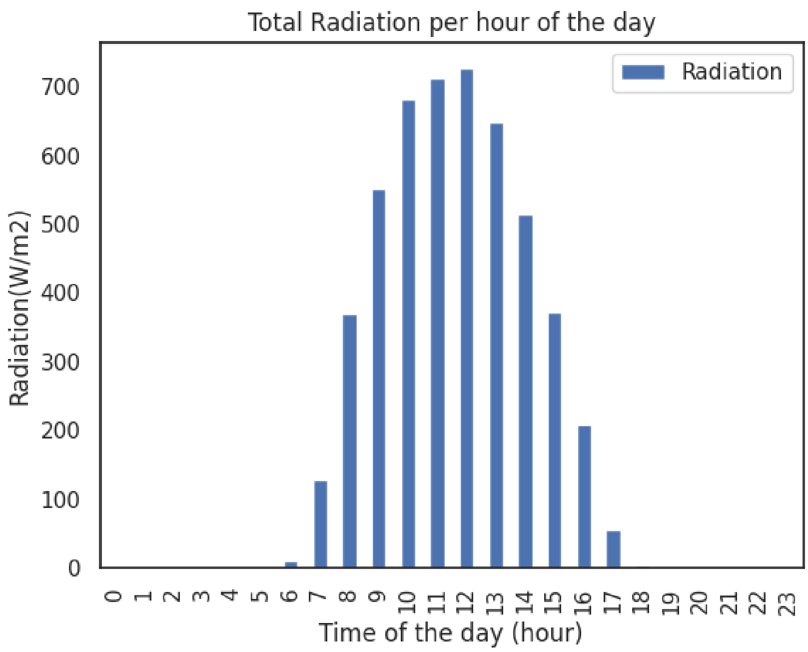


Figure 4.6: Graph plot of solar radiation against hours in a day after taking the hourly mean of the dataset.

The figure 4.6 shows that sun rises at 6am and sun sets at 5pm while the highest radiation is at 12 noon. As expected, both the sun's irradiance and temperature reach their highest point at noon.

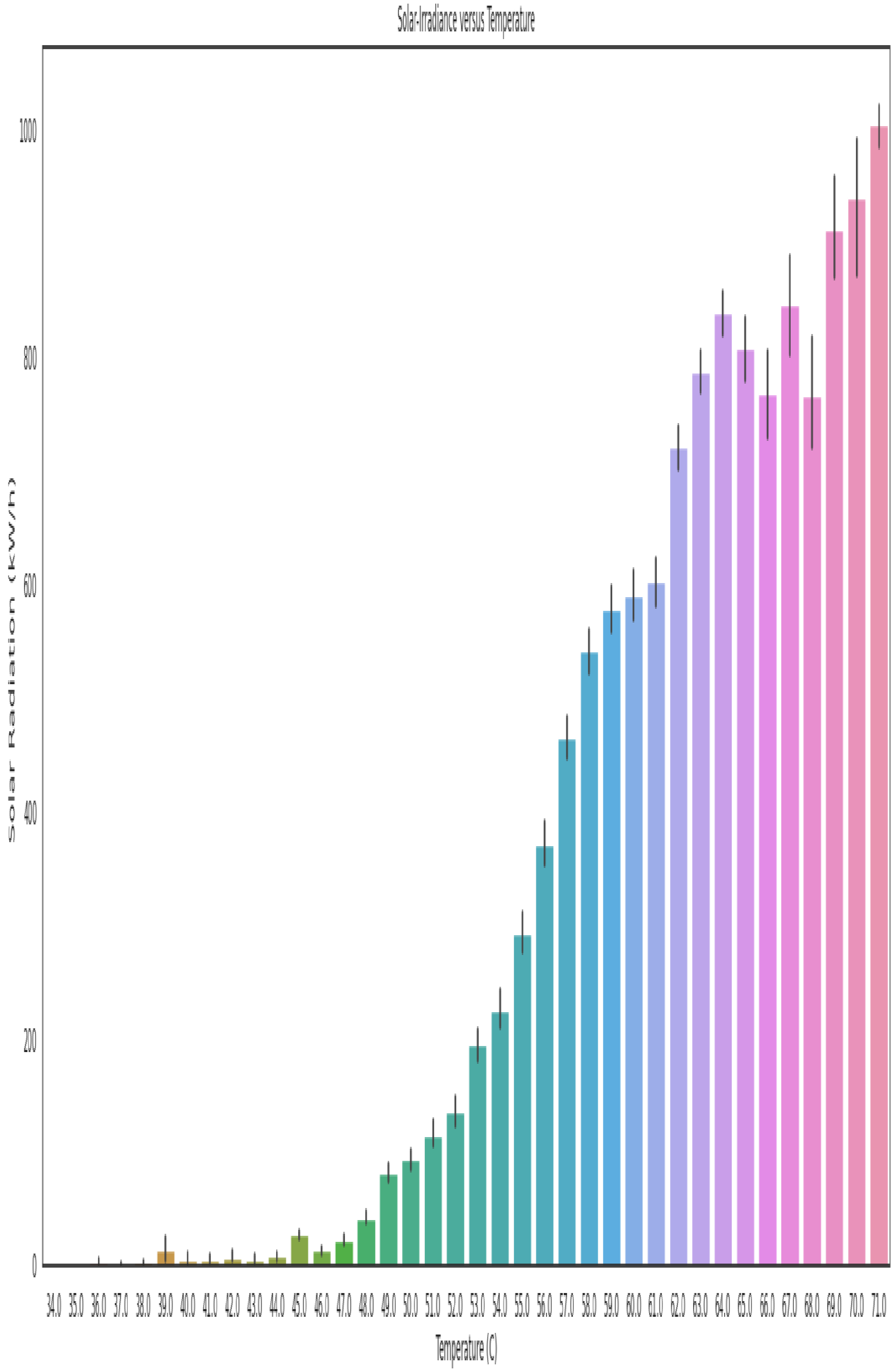


Figure 4.7: Graph Plot solar radiation against temperature

4.2 Transformer Results and Discussion

This is shared in the google link drive and the attached excel sheet.

METRICS	Transformer
MEAN	- 0.03651
MSE	0.003756
MAE	0.045543
R2	0.976298

Table 4.1: The metric result from Time series Transformer

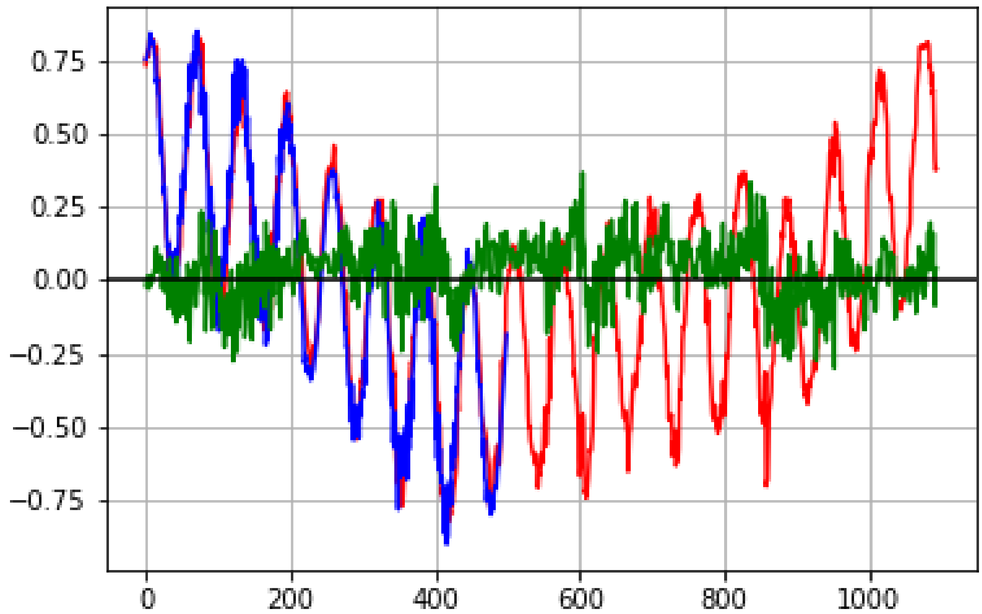


Figure 4.8a: The transformer prediction graph with given dataset for 1093 values at the end of 100 epochs.

In the figure 4.8a, visualization of the predicted values (test_result) in red, true values (truth) in blue, and the difference between predicted and true values in green. i.e. the prediction is color red, the actual ground truth for the first 500 data is plotted in color blue while the transformer is used to plot the difference between predicted and ground truth using test results, in green colour for 1093 values at the end of 100 epochs.

Results are exported to CSV and metrics are calculated with excel sheet to get Mean as -0.03651, MSE as 0.003756, MAE as 0.045543, R^2 as 0.976298.

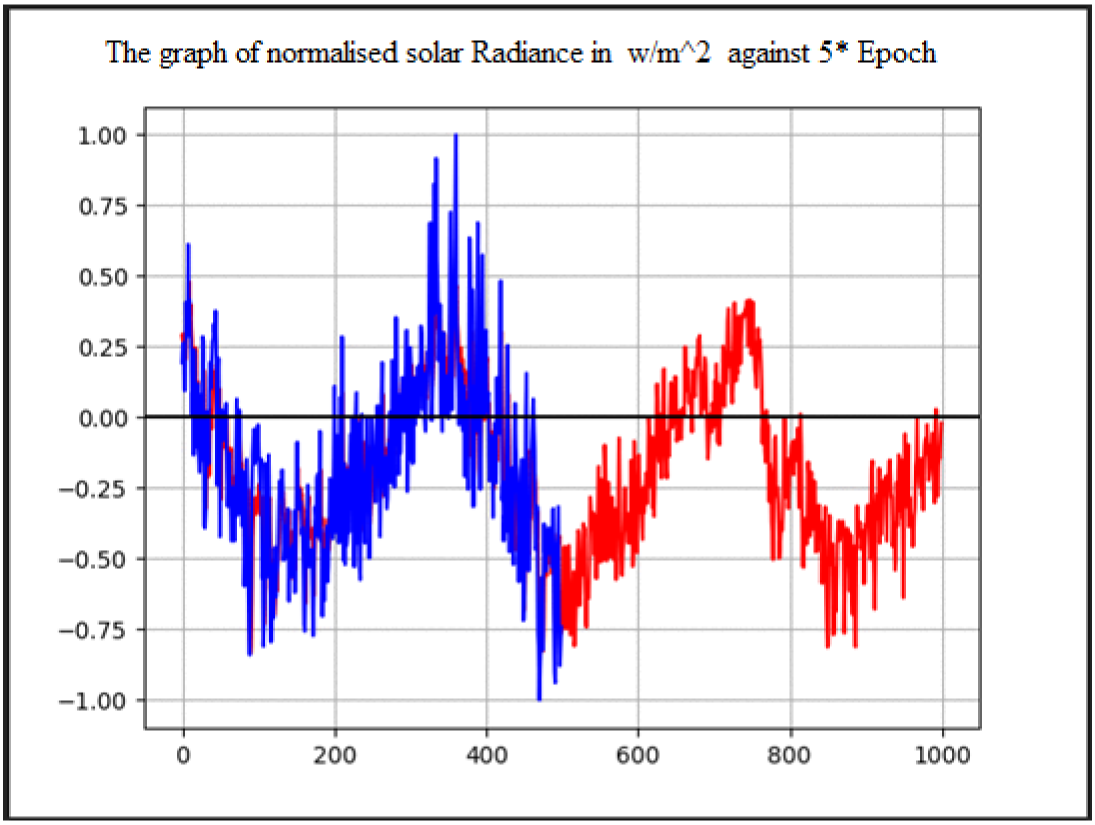
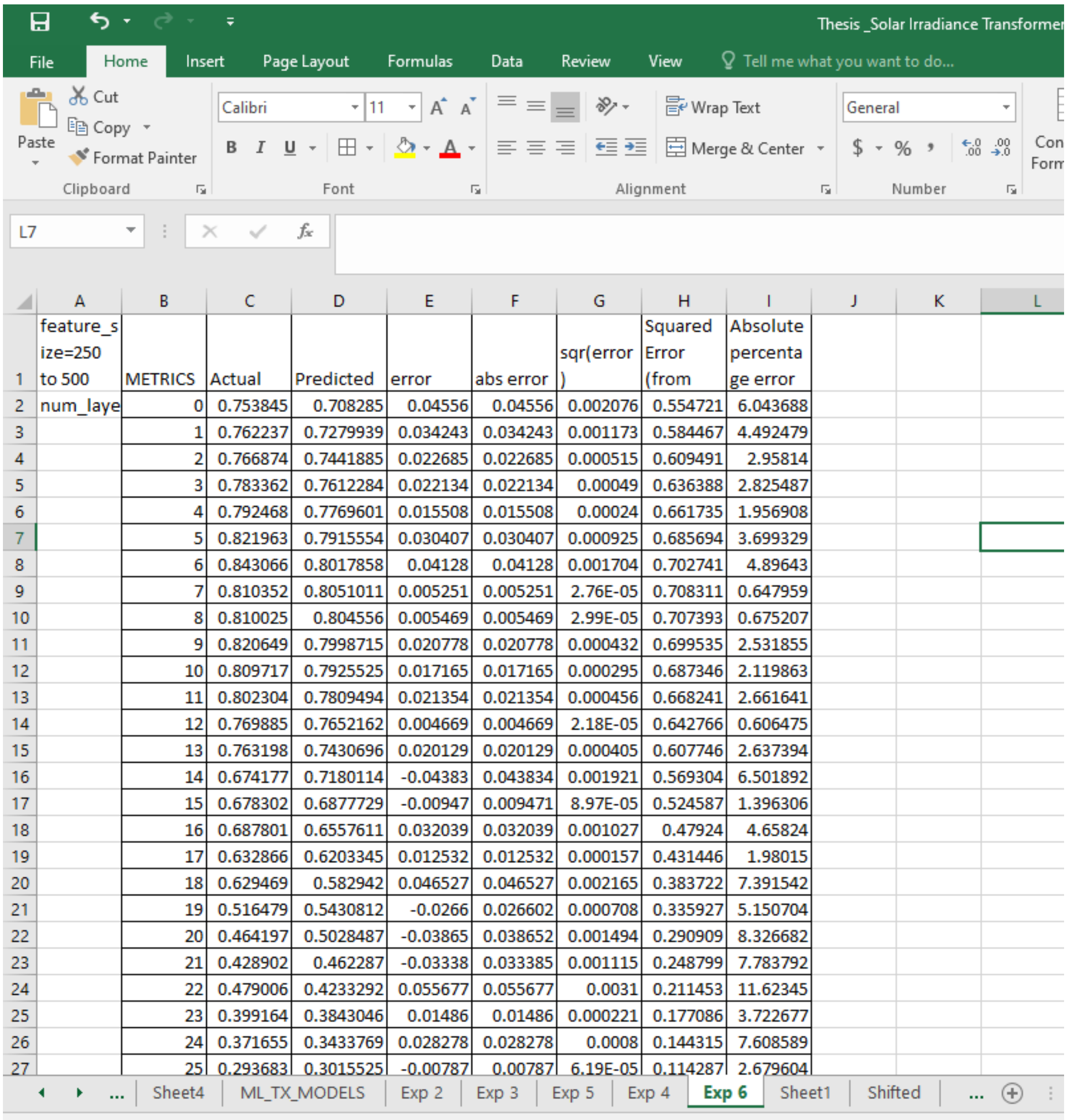


Figure 4.8b: Result output- The normalised solar Radiance against 5* Epochs. The output of a single step prediction model that has been trained for 100 epochs on the NASA Kaggle dataset.

In figure 4.8b, the result output of a single step prediction model that has been trained for 100 epochs on the NASA dataset, input is blue and prediction is red. The predict_future function takes the trained eval_model, data_source, and steps as inputs. It predicts future values by iterating steps times and appending the model's predictions to the data tensor. The resulting data tensor is plotted, with the original data in blue and the predicted future values in red



	A	B	C	D	E	F	G	H	I
1	feature_size=250 to 500	METRICS	Actual	Predicted	error	abs error	sqr(error)	Squared Error (from	Absolute percentage error
2	num_laye	0	0.753845	0.708285	0.04556	0.04556	0.002076	0.554721	6.043688
3		1	0.762237	0.7279939	0.034243	0.034243	0.001173	0.584467	4.492479
4		2	0.766874	0.7441885	0.022685	0.022685	0.000515	0.609491	2.95814
5		3	0.783362	0.7612284	0.022134	0.022134	0.00049	0.636388	2.825487
6		4	0.792468	0.7769601	0.015508	0.015508	0.00024	0.661735	1.956908
7		5	0.821963	0.7915554	0.030407	0.030407	0.000925	0.685694	3.699329
8		6	0.843066	0.8017858	0.04128	0.04128	0.001704	0.702741	4.89643
9		7	0.810352	0.8051011	0.005251	0.005251	2.76E-05	0.708311	0.647959
10		8	0.810025	0.804556	0.005469	0.005469	2.99E-05	0.707393	0.675207
11		9	0.820649	0.7998715	0.020778	0.020778	0.000432	0.699535	2.531855
12		10	0.809717	0.7925525	0.017165	0.017165	0.000295	0.687346	2.119863
13		11	0.802304	0.7809494	0.021354	0.021354	0.000456	0.668241	2.661641
14		12	0.769885	0.7652162	0.004669	0.004669	2.18E-05	0.642766	0.606475
15		13	0.763198	0.7430696	0.020129	0.020129	0.000405	0.607746	2.637394
16		14	0.674177	0.7180114	-0.04383	0.043834	0.001921	0.569304	6.501892
17		15	0.678302	0.6877729	-0.00947	0.009471	8.97E-05	0.524587	1.396306
18		16	0.687801	0.6557611	0.032039	0.032039	0.001027	0.47924	4.65824
19		17	0.632866	0.6203345	0.012532	0.012532	0.000157	0.431446	1.98015
20		18	0.629469	0.582942	0.046527	0.046527	0.002165	0.383722	7.391542
21		19	0.516479	0.5430812	-0.0266	0.026602	0.000708	0.335927	5.150704
22		20	0.464197	0.5028487	-0.03865	0.038652	0.001494	0.290909	8.326682
23		21	0.428902	0.462287	-0.03338	0.033385	0.001115	0.248799	7.783792
24		22	0.479006	0.4233292	0.055677	0.055677	0.0031	0.211453	11.62345
25		23	0.399164	0.3843046	0.01486	0.01486	0.000221	0.177086	3.722677
26		24	0.371655	0.3433769	0.028278	0.028278	0.0008	0.144315	7.608589
27		25	0.293683	0.3015525	-0.00787	0.00787	6.19E-05	0.114287	2.679604

Figure 4.9 showing the data values of ground truth(Actual) and the predicted. Which was plotted in Figure 4.8a but values are exported as Csv then processed for each experiment in order to obtain optimized hyper parameter for TST Transformer.

After this, we essentially train the time series transformer model for the specified number of epochs (100 epoch), evaluates its performance on the validation data, and saves the best model based on the validation loss. Additionally, it generates plots and predicts future values at regular intervals during training.

Untitled - Notepad

File Edit Format View Help

```
| epoch   1 |    32/   84 batches | lr 0.000100 | 1274.68 ms | loss 0.15023 | ppl     1.16
| epoch   1 |    48/   84 batches | lr 0.000100 | 1187.56 ms | loss 0.09714 | ppl     1.10
| epoch   1 |    64/   84 batches | lr 0.000100 | 1235.99 ms | loss 0.09799 | ppl     1.10
| epoch   1 |    80/   84 batches | lr 0.000100 | 1193.80 ms | loss 0.10347 | ppl     1.11
-----------------------------------------------------------------------------------------
| end of epoch   1 | time: 119.46s | valid loss 2.19668 | valid ppl     9.00
-----------------------------------------------------------------------------------------
| epoch   2 |    16/   84 batches | lr 0.000096 | 1266.77 ms | loss 0.67250 | ppl     1.96
| epoch   2 |    32/   84 batches | lr 0.000096 | 1219.74 ms | loss 0.59360 | ppl     1.81
| epoch   2 |    48/   84 batches | lr 0.000096 | 1203.71 ms | loss 0.09073 | ppl     1.09
| epoch   2 |    64/   84 batches | lr 0.000096 | 1290.72 ms | loss 0.03693 | ppl     1.04
| epoch   2 |    80/   84 batches | lr 0.000096 | 1220.29 ms | loss 0.37890 | ppl     1.46
-----------------------------------------------------------------------------------------
| end of epoch   2 | time: 118.40s | valid loss 1.93547 | valid ppl     6.93
-----------------------------------------------------------------------------------------
| epoch   3 |    16/   84 batches | lr 0.000094 | 1308.52 ms | loss 0.48482 | ppl     1.62
| epoch   3 |    32/   84 batches | lr 0.000094 | 1203.66 ms | loss 1.02099 | ppl     2.78
| epoch   3 |    48/   84 batches | lr 0.000094 | 1218.48 ms | loss 0.49285 | ppl     1.64
| epoch   3 |    64/   84 batches | lr 0.000094 | 1211.19 ms | loss 0.65825 | ppl     1.93
| epoch   3 |    80/   84 batches | lr 0.000094 | 1222.43 ms | loss 1.44755 | ppl     4.25
-----------------------------------------------------------------------------------------
| end of epoch   3 | time: 117.72s | valid loss 0.27151 | valid ppl     1.31
-----------------------------------------------------------------------------------------
| epoch   4 |    16/   84 batches | lr 0.000092 | 1303.56 ms | loss 0.20711 | ppl     1.23
| epoch   4 |    32/   84 batches | lr 0.000092 | 1187.31 ms | loss 0.21065 | ppl     1.23
| epoch   4 |    48/   84 batches | lr 0.000092 | 1216.39 ms | loss 0.36784 | ppl     1.44
| epoch   4 |    64/   84 batches | lr 0.000092 | 1206.79 ms | loss 0.59344 | ppl     1.81
| epoch   4 |    80/   84 batches | lr 0.000092 | 1219.04 ms | loss 0.95486 | ppl     2.60
-----------------------------------------------------------------------------------------
| end of epoch   4 | time: 117.27s | valid loss 0.36198 | valid ppl     1.44
-----------------------------------------------------------------------------------------
| epoch   5 |    16/   84 batches | lr 0.000090 | 1335.39 ms | loss 0.24084 | ppl     1.27
| epoch   5 |    32/   84 batches | lr 0.000090 | 1197.59 ms | loss 0.14879 | ppl     1.16
| epoch   5 |    48/   84 batches | lr 0.000090 | 1237.26 ms | loss 0.24497 | ppl     1.28
| epoch   5 |    64/   84 batches | lr 0.000090 | 1201.20 ms | loss 0.59352 | ppl     1.81
| epoch   5 |    80/   84 batches | lr 0.000090 | 1252.86 ms | loss 0.55039 | ppl     1.73
-----------------------------------------------------------------------------------------
| end of epoch   5 | time: 118.32s | valid loss 0.21686 | valid ppl     1.24
-----------------------------------------------------------------------------------------
| epoch   6 |    16/   84 batches | lr 0.000089 | 1333.43 ms | loss 0.24373 | ppl     1.28
| epoch   6 |    32/   84 batches | lr 0.000089 | 1211.31 ms | loss 0.36464 | ppl     1.44
| epoch   6 |    48/   84 batches | lr 0.000089 | 1259.08 ms | loss 0.26853 | ppl     1.31
| epoch   6 |    64/   84 batches | lr 0.000089 | 1221.23 ms | loss 0.18277 | ppl     1.20
| epoch   6 |    80/   84 batches | lr 0.000089 | 1259.96 ms | loss 0.26069 | ppl     1.30
```

Figure 4.10 shows output of the epoch number, time taken, validation loss, and validation perplexity are printed.

This is what was happening in the TST transformer prediction process when using the training and testing data. The predict_future function takes the trained eval_model, data_source, and steps as inputs. It predicts future values by iterating steps times and appending the model's predictions to the data tensor. The resulting data tensor is plotted, with the original data in blue and the predicted future values in red. The plot is saved as an image in Figure 4.8b

The evaluate function evaluates the trained eval_model on the given data_source. It calculates the loss between the model's predictions and the target values, accumulates the loss in total_loss, and returns the average loss per data point.

The code initializes the training and validation data using the get_data function and creates an instance of the TransAm transformer model. The loss criterion is defined as the Mean Squared Error (MSELoss) using nn.MSELoss(). The learning rate (lr) and optimizer (Adam) are defined. A learning rate scheduler is created using torch.optim.lr_scheduler.StepLR, which applies a step decay to the learning rate.

The variables best_val_loss and best_model are initialized to track the best validation loss and the corresponding best model. The main training loop starts, iterating over the specified number of epochs. The training data is passed to the train function to train the model for one epoch. If the current epoch is a multiple of 10, the validation loss is computed by calling plot_and_loss to generate a plot of predicted vs. true values and predict_future to predict future values. If the current epoch is not a multiple of 10, the validation loss is computed using the evaluate function. The epoch number, time taken, validation loss, and validation perplexity are printed. If the current validation loss is better than the previous best validation loss, the current model is saved as the best model. The learning rate scheduler is updated in figure 4.10 Here the set was The get_data function is defined to generate the solar irradiance data for training and testing the model. It creates a time array time ranging from 0 to 400 with a step of 0.1.The amplitude array is computed by adding the sine waves with different frequencies and amplitudes. Gaussian noise is also added to the data. The MinMaxScaler from sklearn.preprocessing is used to scale the amplitude values between -1 and 1. This is why the Figure 4.11 is a sinusoidal wave.

The code essentially trains the time series transformer model for the specified number of epochs, evaluates its performance on the validation data, and saves the best model based on the validation loss. Additionally, it generates plots and predicts future values at regular intervals during training. At 100 Ephoc the figure 4.8a is generated and at that instance, Figure 4.8b represents the graph single step prediction which uses length of output_window which is 5 and not one prediction. In order to adopt to one prediction, median or mean of the range of the values of output-window can be used.

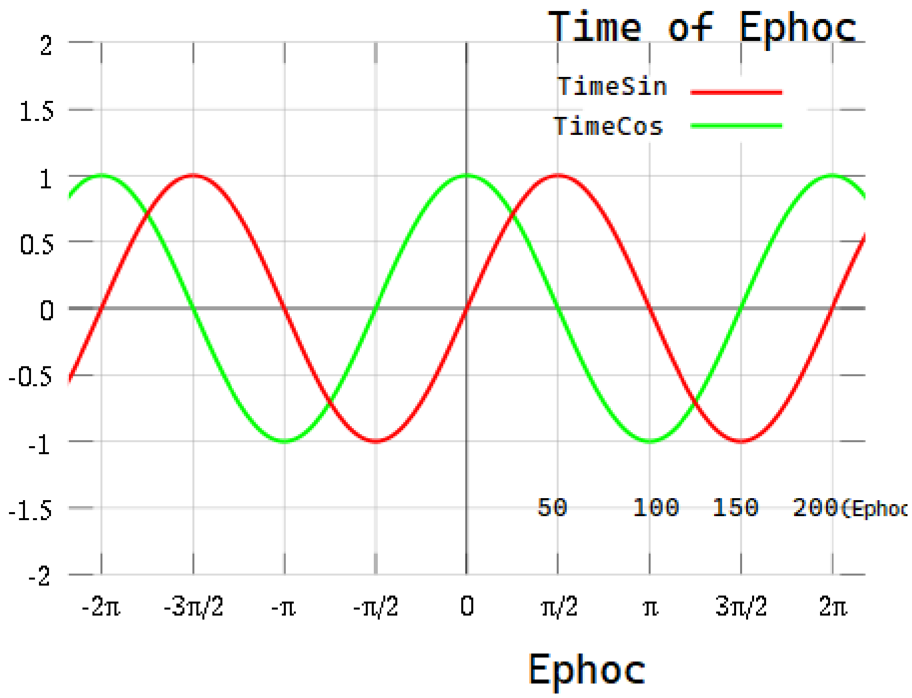


Figure 4.11: Visualization of visible correlations between number of Ephoc and normalised solar irradiance data.

Hence in other for us to use Probabilistic Time Series Forecasting with Transformers, the mean/ median prediction values are referenced against actual values since it is Probabilistic Time Series Forecasting Transformers and are not used for one prediction.

Since in Figure 4.11, The values moves between -1 and 1, a complete cycle is 2π = 1 Ephoc

TimeSin = $\frac{\text{Sin } 2\pi \ Time}{Ephoc}$

TimeCos = $\frac{\text{Cos } 2\pi \ Time}{Ephoc}$

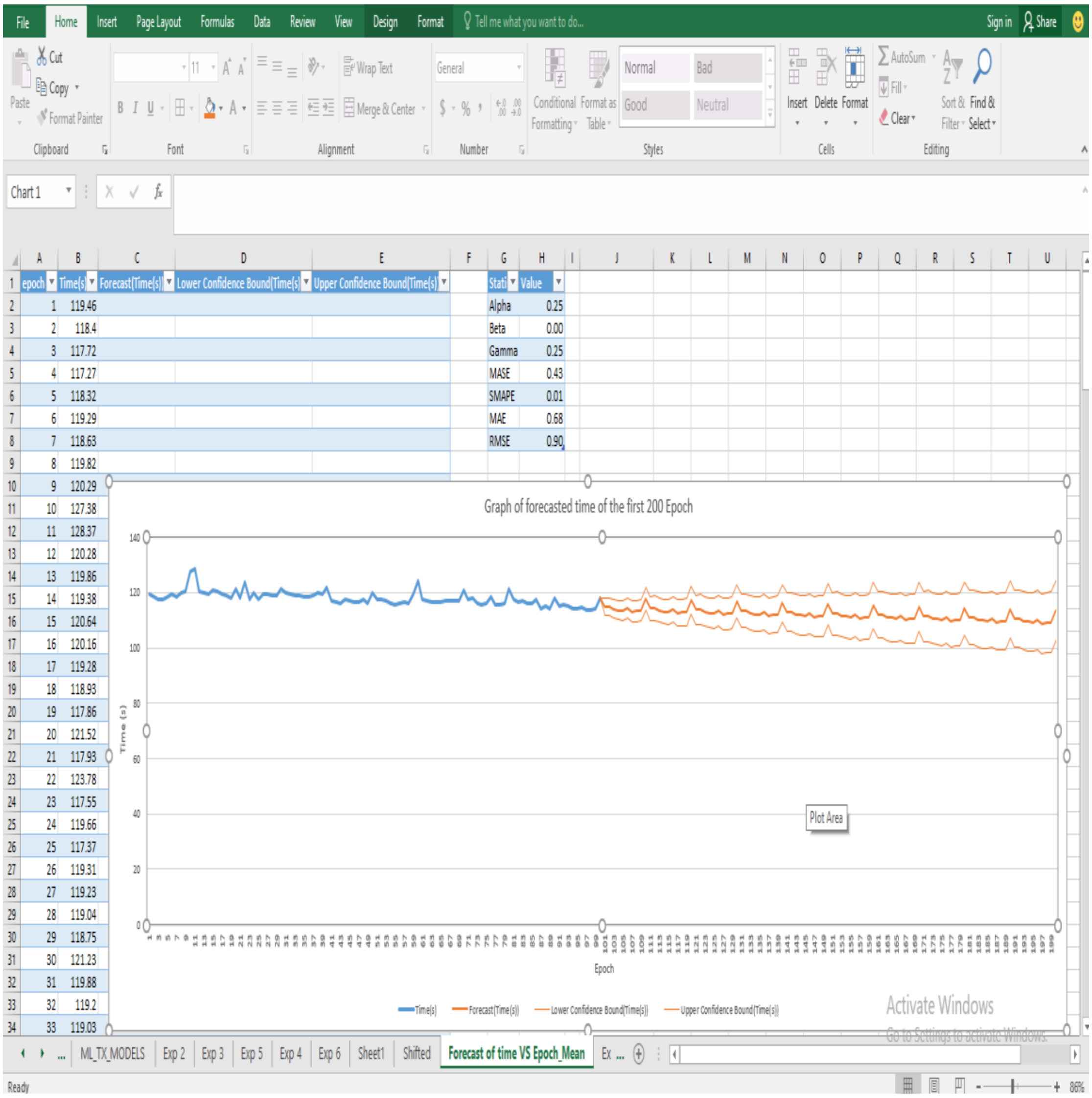


Figure 4.12: Visualization of Actual time spent on the first 100 Ephoc and forecasted time for the next 100 Ephoc

4.3 ML and Transformer Discussion

The model is trained to make predictions of solar irradiance at future time steps using input data consisting of solar radiation measurements over time. The architecture of the model is defined within the TransAm class constructor, where the layers and their respective numbers are set up. The forward method is then utilized to pass the data through the model in a forward direction. This involves processing the input data through the positional encoder, transformer encoder, and decoder layers to generate the final output.

Several hyper parameters are employed in the training process, such as an input window size of 100, an output window size of 5, and a batch size of 10. The transformer model performs well when trained on the raw dataset, yielding satisfactory results with a small mean value of -0.03651, a mean squared error (MSE) of 0.003756, a mean absolute error (MAE) of 0.045543, and an R-squared (R^2) value of 0.976298.

4.4 Major performance, achievement and result analysis

Utilizing the technique of "teacher forcing" is an essential aspect of the experimental setup in the transformer architecture, as it plays a critical role in training the model weights. The concept involves feeding the ground-truth sequence values back into the time series transformer at each step, forcing the model to align closely with the actual sequence. Analogous to a student taking a multi-part exam, where each answer depends on the previous one, "teacher forcing" provides immediate feedback and correct answers to guide the model's learning.

The model utilizes a training technique known as "teacher-forcing," which is commonly used in training Transformers for machine translation tasks. In this approach, during training, the future values are shifted one position to the right and fed as input to the decoder, with the last value of the past values appended. At each time step, the model is tasked with predicting the subsequent target value. The training setup resembles that of a GPT (Generative Pre-trained Transformer) model for language, although there is no concept of a decoder_start_token_id. Instead, the last value of the context is used as the initial input for the decoder.

During inference, the decoder is provided with the final value of the past values as input. Subsequently, the model can sample from its learned distribution to generate a prediction for the next time step. This prediction is then fed back into the decoder to generate the subsequent prediction (Kashif, 2022).

Training with teacher forcing offers several advantages. It enables faster convergence of the model by preventing error accumulation during the initial stages when predictions may be inaccurate. Without teacher forcing, incorrect predictions would update the model's hidden states, leading to error accumulation and hindered learning. Additionally, teacher forcing helps stabilize the training process by preventing error propagation caused by incorrect previous outputs in the generated sequence. Moreover, in certain sequence-to-sequence tasks, using teacher forcing can yield better performance compared to training without it.

In this study, various experimental parameters were optimized for the time series transformer model, including the number of layers, neurons, learning rate, batch size, and epochs during training. The main focus was to assess the performance of the time series transformer model, so the parameters were fine-tuned to optimize its effectiveness. Other 10 machine learning models, such as those from the Sklearn library, were also considered, and their parameters were adjusted based on their specific characteristics from the library. The optimized parameter values obtained through experimentation are summarized in Table 4.2.

Method	Hyper Parameters	Values
TST Transformer	Number of epochs	100
	Feature size	500
	Learning rate	0.0001
	Batch size	32
	Optimizer	Adam
	Input window	100
	Output window	5
	Number of layers	2
10 other ML models	Sklearn library parameter	

Table 4.2 Comparison between Models from Experimental Results

To put things into perspective. We summarize the performance metrics obtained for the best models in Table 4.4 where we conclude that the transformer is the best model we got.

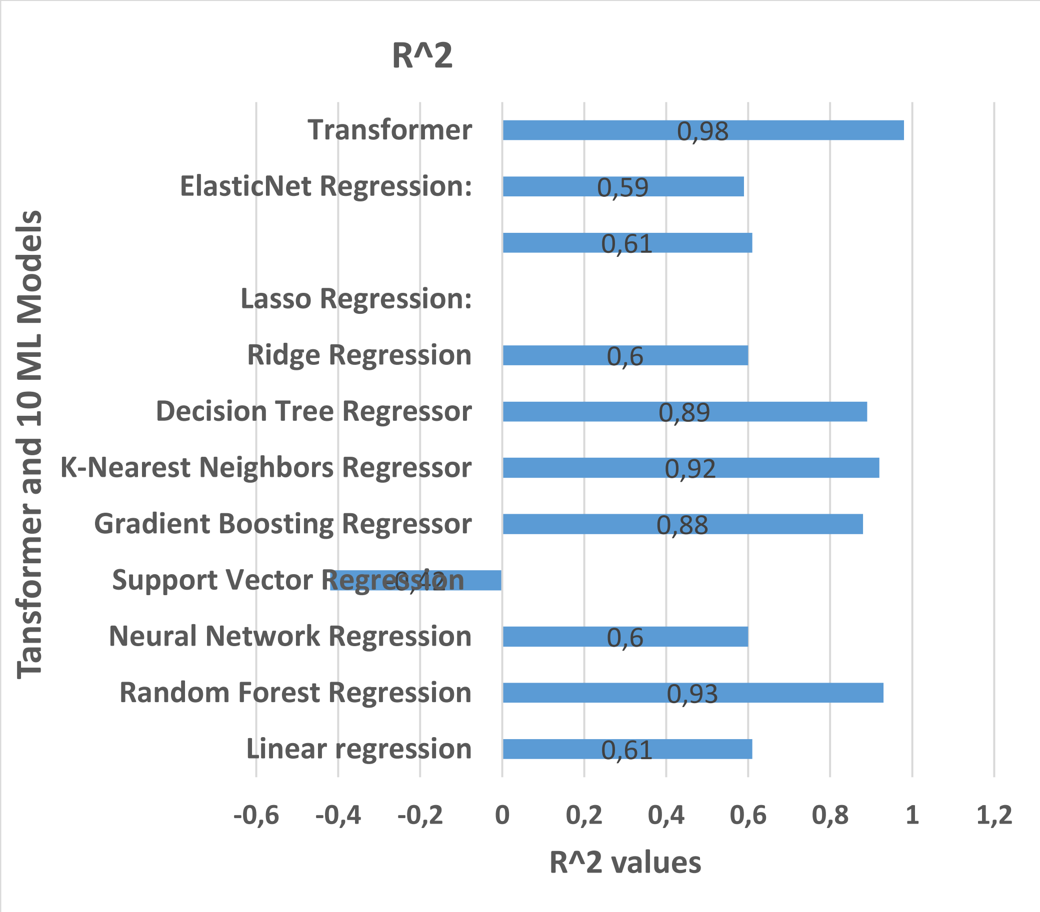


Figure 4.13: R^2 value of the Transformer and 10 other Machine learning models

METRICS	MSE	RMSE	MAE	R^2
Linear regression.	38372.19	195.89	148.53	0.61
Random Forest Regression	6486.25	80.54	30.87	0.93
Neural Network Regression	40131.84	200.33	152.87	0.60
Support Vector Regression	144548.79	380.20	207.91	-0.42
Gradient Boosting Regressor	11964.49	109.38	59.97	0.88
K-Nearest Neighbors Regressor	8140.82	90.23	38.21	0.92
Decision Tree Regressor	10536.45	102.65	37.21	0.89
Ridge Regression	39957.61	199.89	152.93	0.60
Lasso Regression:	37810.72	194.45	148.97	0.61
ElasticNet Regression:	40756.89	201.88	152.66	0.59
* Transformer	0.00375	0.06	0.045543	0.976

Table 4.4: Performance Metrics on the testing set for the best models. Transformer model used normalised dataset.

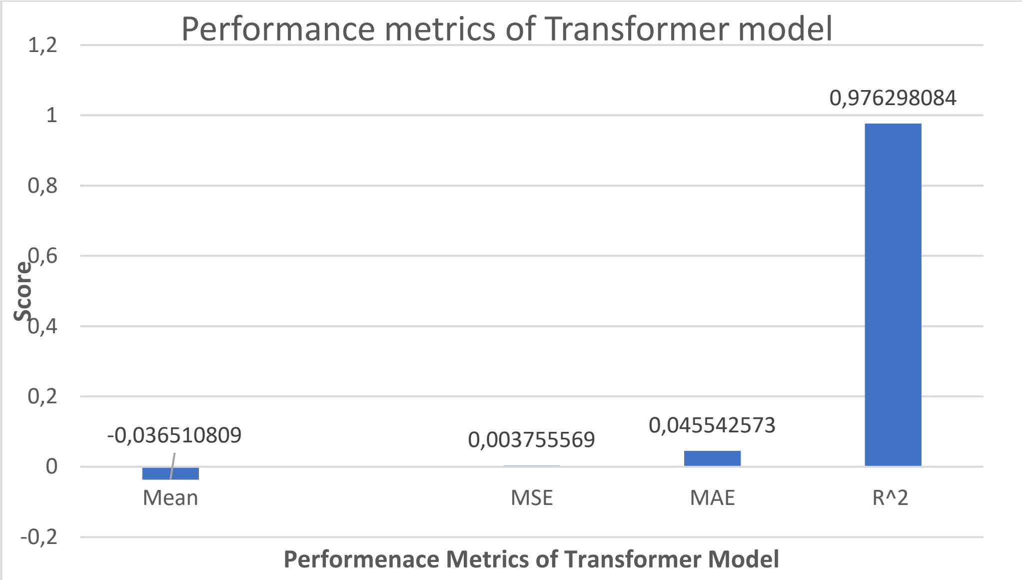


Figure 4.14: Bar plot of Mean, MSE, MAE and R^2 of Transformer training process.

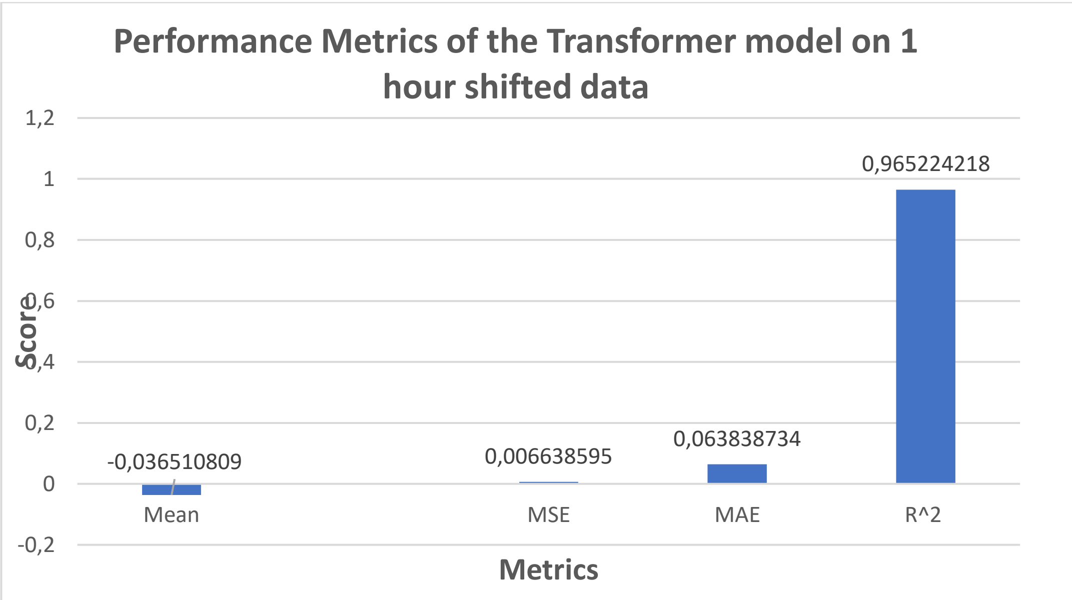


Figure 4.15 Bar plot of Mean, MSE, MAE and R^2 of Transformer training process on 1 hour shifted data.

Paper	Location	MAPE (%)
[28]	Abha (Saudi Arabia)	11.8
[29]	Sirt (Turkey)	6.78
[30]	Mugla (Turkey)	6.73
[31]	Cyprus and USA	4.7
[32]	Mumbai (India)	4.24
[33]	Chennai Metropolitan Area (India)	3.45
This study	USA	0.68

Table 4.5. Mean absolute percentage error results of other authors for Solar Energy forecasting.

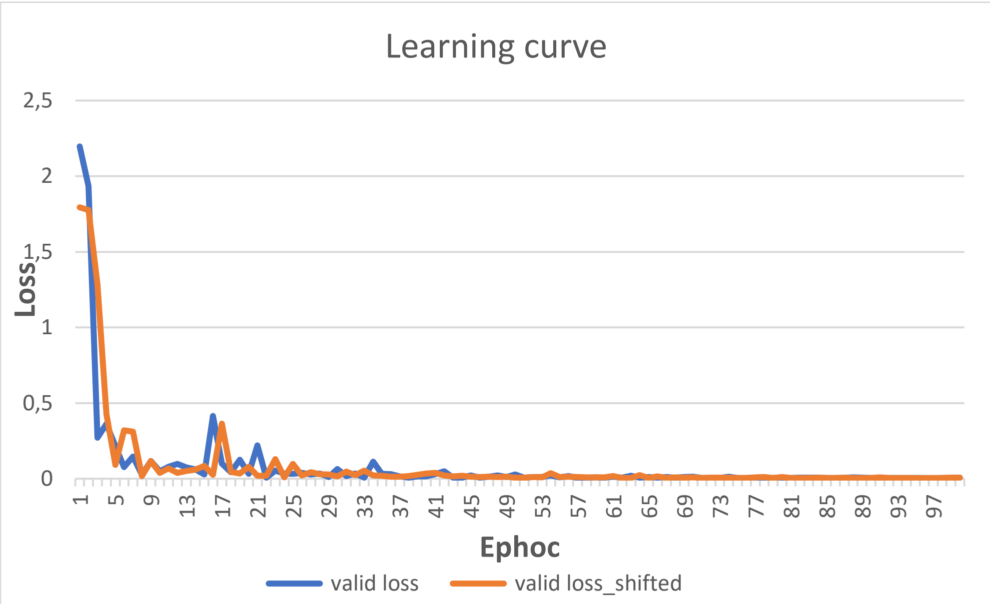
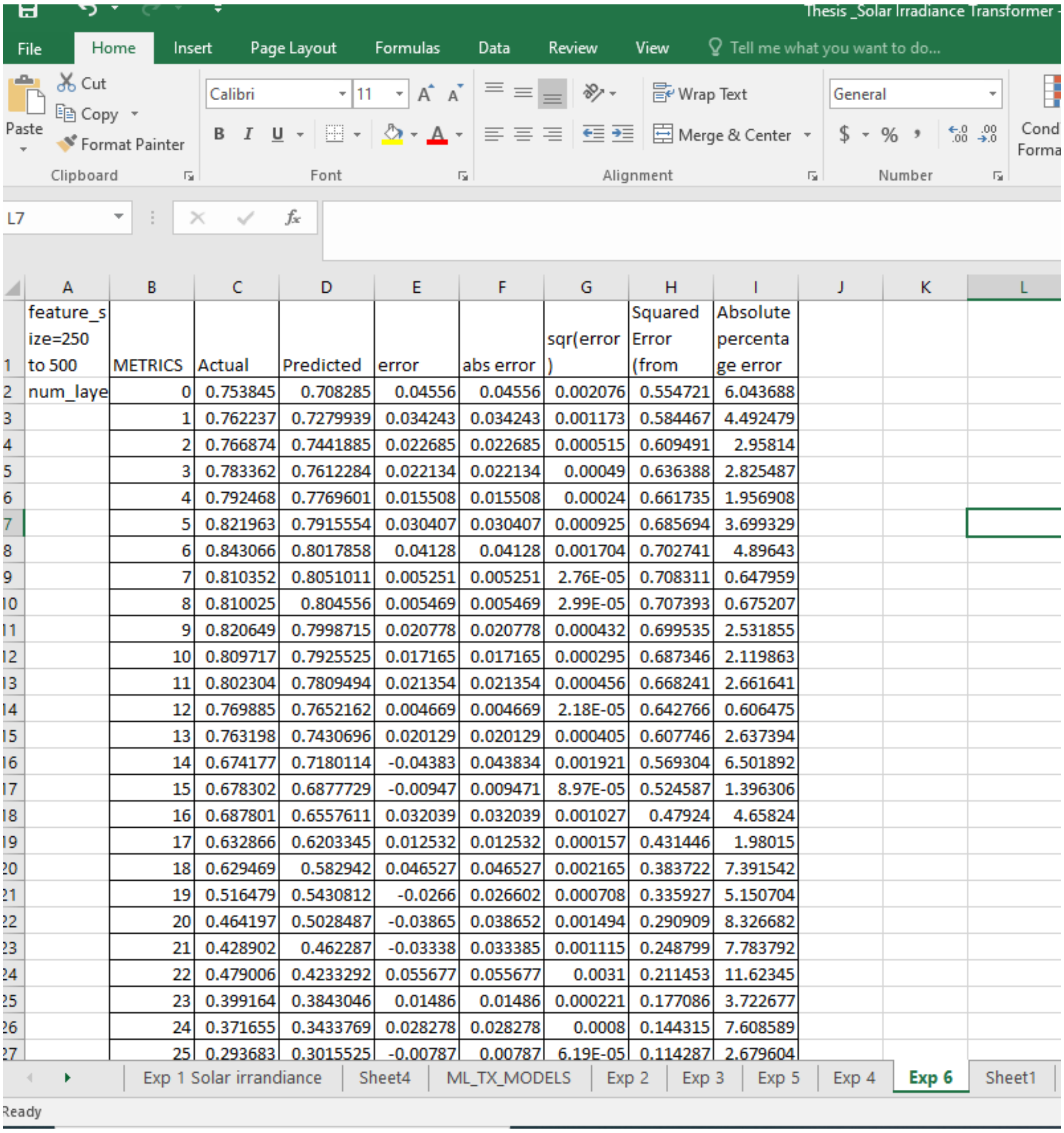


Figure 4.16: *The learning curve of Transformer model*



	A	B	C	D	E	F	G	H	I
1	feature_size=250 to 500	METRICS	Actual	Predicted	error	abs error	sqr(error)	Squared Error (from	Absolute percentage error
2	num_laye	0	0.753845	0.708285	0.04556	0.04556	0.002076	0.554721	6.043688
3		1	0.762237	0.7279939	0.034243	0.034243	0.001173	0.584467	4.492479
4		2	0.766874	0.7441885	0.022685	0.022685	0.000515	0.609491	2.95814
5		3	0.783362	0.7612284	0.022134	0.022134	0.00049	0.636388	2.825487
6		4	0.792468	0.7769601	0.015508	0.015508	0.00024	0.661735	1.956908
7		5	0.821963	0.7915554	0.030407	0.030407	0.000925	0.685694	3.699329
8		6	0.843066	0.8017858	0.04128	0.04128	0.001704	0.702741	4.89643
9		7	0.810352	0.8051011	0.005251	0.005251	2.76E-05	0.708311	0.647959
10		8	0.810025	0.804556	0.005469	0.005469	2.99E-05	0.707393	0.675207
11		9	0.820649	0.7998715	0.020778	0.020778	0.000432	0.699535	2.531855
12		10	0.809717	0.7925525	0.017165	0.017165	0.000295	0.687346	2.119863
13		11	0.802304	0.7809494	0.021354	0.021354	0.000456	0.668241	2.661641
14		12	0.769885	0.7652162	0.004669	0.004669	2.18E-05	0.642766	0.606475
15		13	0.763198	0.7430696	0.020129	0.020129	0.000405	0.607746	2.637394
16		14	0.674177	0.7180114	-0.04383	0.043834	0.001921	0.569304	6.501892
17		15	0.678302	0.6877729	-0.00947	0.009471	8.97E-05	0.524587	1.396306
18		16	0.687801	0.6557611	0.032039	0.032039	0.001027	0.47924	4.65824
19		17	0.632866	0.6203345	0.012532	0.012532	0.000157	0.431446	1.98015
20		18	0.629469	0.582942	0.046527	0.046527	0.002165	0.383722	7.391542
21		19	0.516479	0.5430812	-0.0266	0.026602	0.000708	0.335927	5.150704
22		20	0.464197	0.5028487	-0.03865	0.038652	0.001494	0.290909	8.326682
23		21	0.428902	0.462287	-0.03338	0.033385	0.001115	0.248799	7.783792
24		22	0.479006	0.4233292	0.055677	0.055677	0.0031	0.211453	11.62345
25		23	0.399164	0.3843046	0.01486	0.01486	0.000221	0.177086	3.722677
26		24	0.371655	0.3433769	0.028278	0.028278	0.0008	0.144315	7.608589
27		25	0.293683	0.3015525	-0.00787	0.00787	6.19E-05	0.114287	2.679604

Figure 4.17 The Excel calculations and experimental models documentation.

The TST Transformer gave output result data i.e. 1093 Actual Values and 1093 predicted values after it has been trained with NASA Kaggle Dataset on several experiments which investigating for optimized hyper parameter values on the TST Transformer model.

4.5 Real-time prediction on a recent weather data and business use case

Technology advances in recent years have completely changed how renewable energy is produced and used. Researchers and companies now have access to extremely exact wind and solar power data because to the power of AI, computer power, and more accurate weather forecasts. Initiatives aiming at improving the effectiveness of renewable energy systems have increased sharply as a result. For instance, Google signs 140MW energy contract with Engie in Germany. Google has used machine learning to enhance predictions of wind output and has intentionally planned computing jobs to coincide with times when solar and wind power are most abundant, optimizing the use of renewable resources. The performance of solar panels and the influence of clouds have been predicted using a unique technique, which is important because solar energy systems also depend on weather conditions. (Engie, 2020)

CHAPTER V

5.0 Conclusion

Transformers allow us to train very large models since they are highly parallelizable, exceedingly compute-optimal, and efficient. This is one of the main differences between transformers and other designs. Transformers were used to solve this time series issue, and the results were favourable in terms of MSE, MAE, and R2. Recent research has focused on solar irradiance prediction because of the demand for and interest in green and renewable energy. Accurate solar irradiance forecasting that takes into consideration both potential and forecasting challenges is necessary to fully grasp the solar energy viewpoint of a site. Solar irradiance data may be efficiently and precisely predicted using Transformer models. The time series Transformer models were employed in this study to predict the solar irradiance data, and they provided an effective and accurate forecast of solar irradiance when compared to other machine models on the NASA Dataset. MSE, MAE, and coefficients of determination(R2) were used to verify the model's forecasting accuracy, goodness of fit in order to validate and stabilize the simulation findings. TST performed best in all performance metrics. The outcomes showed that the suggested approach was capable of making precise predictions of solar irradiance. MSE is 0.003756, MAE is 0.045543 ,97.6% is the value of the R2 coefficient of determination. MAPE is 0.68% which is lower than that of the state of the art.

5.1 Future Works

This thesis explores the application of transformer for solar irradiance forecasting in the energy sector. Transformer has demonstrated good results in time series forecasting, and by applying them to predict solar irradiance, they can assist to optimize energy in the digital world.

In this study, we present the transformer model, a solar irradiance forecasting model based on encoder-decoder technology. In further research, Feature selection will be used in the transformer net architecture to enhance the standard Transformer encoder for encoder-decoder-based long-term prediction

5.2 Recommendations

Studies also show that the energy needed to train AI models on increasing a reliable and economic integration of PV by grid operators and to optimize energy consumption management are minimal to energy saved for their use, which is important in terms of the sustainability of AI technologies in the energy system. The use of intelligent optimization and AI technology in the energy system is therefore quite sensible. The optimistic hopes people have, however, are overstated (looking at the big picture of AI solving every problem) since there are still obstacles that AI cannot overcome, such as the creation of an acceptable regulatory framework or the involvement of citizens in the energy system design and more especially citizen's participation in the design of AI system itself. The fact is that a lot of pilot initiatives fail because of the regulatory environment. In order for AI to genuinely contribute to improving the world, certain conditions must be created concurrently with the advancement of the technology i.e. AI designs must be engineered to be rights respecting, inclusive, safe and trusted tool for all. Sustainability, empowerment, security and freedom should be at the centre of the digital transformation (including AI).

This research highlights the incorporation of date-time data alongside normalized solar radiation time series. While NLP and Vision fields have benefited from pre-trained transformer models, the time series domain remains relatively unexplored in this regard. The study suggests that Transformer-based models hold great promise for advancing time series analysis, and researchers are encouraged to explore this area further.

REFERENCES

Abdel-Nasser, M., Mahmoud, K., & Lehtonen, M. (2020). Reliable solar irradiance forecasting approach based on choquet integral and deep LSTMs. *IEEE Transactions on Industrial Informatics, 17(3),* 1873-1881. Article (accessed on 12 Jan. 2023).

Acheampong FA, Nunoo-Mensah H, Chen W (2021) Transformer models for text-based emotion detection: a review of BERT-based approaches. *Artif Intell Rev 54(8):*5789–5829. Article Google Scholar (accessed on 12 Jan. 2023).

Akanni.O. et al, Reviewing applications of AI and Blockchain in energy industry, 2023, in print

Alharbi, F.R.; Csala, D. Wind Speed and Solar Irradiance Prediction Using a Bidirectional Long Short-Term Memory Model Based on Neural Networks. *Energies 2021*, 14, 6501. Article (accessed on 12 June 2023).

Alzahrani, A.; Shamsi, P.; Dagli, C.; Ferdowsi, M. Solar irradiance forecasting using deep neural networks. *Procedia Comput. Sci. 2017*, *114*, 304–313.Article (accessed on 12 June 2023).

Brahma, B., & Wadhvani, R. (2020). Solar irradiance forecasting based on deep learning methodologies and multi-site data. *Symmetry*, *12*(11), 1830 . Google Scholar.Article Available online (accessed on 12 Jan. 2023).

Curceac, S., Ternynck, C., Ouarda, T. B., Chebana, F., & Niang, S. D. (2019). Short-term air temperature forecasting using nonparametric functional data analysis and SARMA models. *Environmental Modelling & Software*, *111*, 394-408.Article (accessed on 12 Jan. 2023).

Duffy, A., & Rogers, M., & Ayompe, L. (2015) Renewable energy and energy efficiency: assessment of projects and policies. John Wiley & Sons ,.Article (accessed on 12 Jan. 2023).

European Commission, "Digitalising the energy system - EU action plan" Article (accessed on 12 June 2023).

Grigsby, J., Wang, Z., Nguyen, N., & Qi, Y. (2021). Long-range transformers for dynamic spatiotemporal forecasting. *arXiv preprint arXiv:2109.12218*. Article. Google Scholar (accessed on 12 Jan. 2023).

Hao, Y., & Tian, C. (2019). A novel two-stage forecasting model based on error factor and ensemble method for multi-step wind power forecasting. Applied energy, 238, 368-383Article (accessed on 12 Jan. 2023).

Huang, X., Li, Q., Tai, Y., Chen, Z., Zhang, J., Shi, J., ... & Liu, W. (2021). Hybrid deep neural model for hourly solar irradiance forecasting. *Renewable Energy*, *171*, 1041-1060.Article (accessed on 12 Jan. 2023).

Husein, M.; Chung, I.Y. Day–ahead solar irradiance forecasting for microgrids using a long short–term memory recurrent neural network*: A deep learning approach. Energies 2019*, 12, 1856.

Available online: https://www.mdpi.com/1996-1073/12/10/1856 Article (accessed on 12 June 2023).

Jlidi, M., Hamidi, F., Barambones, O., Abbassi, R., Jerbi, H., Aoun, M., & Karami-Mollaee, A. (2023). An Artificial Neural Network for Solar Energy Prediction and Control Using Jaya-SMC. *Electronics*, *12(3)*, 592. MDPI AG. http://dx.doi.org/10.3390/electronics12030592 (accessed on 12 June 2023).

Kashif Rasul , huggingface Time series Transformers documentation,(2022, December) Time series Transformers deocumentation Article (accessed on 12 June 2023).

Kumari, P., & Toshniwal, D. (2021). Deep learning models for solar irradiance forecasting A comprehensive review. *Journal of Cleaner Production*, *318*, 128566 Article

Kumari, P., & Toshniwal, D. (2021). Long short term memory–convolutional neural network based deep hybrid approach for solar irradiance forecasting, , *Applied Energy*, 295, 117061 Article (accessed on 12 Jan. 2023

Lai, J. P., Chang, Y. M., Chen, C. H., & Pai, P. F. (2020). A survey of machine learning models in renewable energy predictions. *Applied Sciences*, *10*(17), 5975.Article (accessed on 12 Jan. 2023).

Li, R., Xiao, W., Wang, L., Jang, H., & Carenini, G. (2021, November). T3-vis: visual analytic for training and fine-tuning transformers in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 220-230). Article Publication (accessed on 12 Jan. 2023).

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y. and Yan, X. (2019), Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting, *In NeurIPS 2019*. Article (accessed on 12 June 2023).

Liang, R.H. and Liao, J.H. (2007) A Fuzzy-Optimization Approach for Generation Scheduling with Wind and Solar Energy Systems. *IEEE Transactions on Power Systems, 22,* 1665-1674. Article (accessed on 12 June 2023).

Mendonça de Paiva, G.; Pires Pimentel, S.; Pinheiro Alvarenga, B.; Gonçalves Marra, E.; Mussetta, M.; Leva, S. Multiple site intraday solar irradiance forecasting by machine learning algorithms: MGGP and MLP neural networks. Energies 2020, 13, 3005. .Article (accessed on 12 June 2023).

Middya, A. I., & Roy, S. (2022). Pollutant specific optimal deep learning and statistical model building for air quality forecasting *Environmental Pollution*, *301*, 118972. Article Publication (accessed on 12 Jan. 2023).

Milind Sahay,"Neural Networks and the Universal Approximation Theorem",Jun 6, 2020 , https://towardsdatascience.com/neural-networks-and-the-universal-approximation-theorem-8a389a33d30a (accessed on 12 June 2023

Murata, A., Ohtake, H., & Oozeki, T. (2018). Modeling of uncertainty of solar irradiance forecasts on numerical weather predictions with the estimation of multiple confidence intervals. *Renewable energy*, *117*, 193-201.Article (accessed on 12 Jan. 2023)

NASA Dataset, https://www.kaggle.com/datasets/dronio/SolarEnergy (accessed on 12 Jan. 2023).

NASA Prediction of Worldwide Energy Resource (POWER),Higher Resolution Daily Time Series, Renewable Energy Community Prediction dataset from https://power.larc.nasa.gov/data-access-viewer/ (accessed on 12 June 2023).

Niels Rogge and Kashif Rasul , Probabilistic Time Series Forecasting with Transformers, (2022, December) *Transformer documentation*, , Article (accessed on 12 June 2023).

Nigeria National AI policy: call for contributions call for contributions Article

Pospíchal J, Kubovčík M, Dirgová Luptáková I. Solar Irradiance Forecasting with Transformer Model. *Applied Sciences*. 2022; 12(17):8852. Article (accessed on 12 June 2023)

Prema V, & Rao, U. (2015). Development of statistical time series models for solar power prediction. *Renewable Energy*. *83*. *10.1016/j.renene*.2015.03.038 Article (accessed on 12 June 2023).

Premalatha, N.; Valan Arasu, A. Prediction of solar radiation for solar systems by using ANN models with different back propagation algorithms. J. Appl. Res. Technol. 2016, 14, 206–214. Article (accessed on 12 June 2023).

Pytorch forecasting documentation Article, How to make a Transformer for time series forecasting with PyTorch. Article (accessed on 12 Jan. 2023).

Rehman, S.; Mohandes, M. Artificial neural network estimation of global solar radiation using air temperature and relative humidity. Energy Policy 2008, 36, 571–576. Available online: Article [CrossRef (accessed on 12 June 2023).

Reikard, G. (2009). Predicting solar radiation at high resolutions: A comparison of time series forecasts. *Solar energy*, *83*(3), 342-349.Article (accessed on 12 Jan. 2023).

Sampaio, P. G. V., & González, M. O. A. (2017). Photovoltaic solar energy: Conceptual framework. *Renewable and Sustainable Energy Reviews*, *74*, 590-601. Article, Google Scholar (accessed on 12 Jan. 2023).

Sharma, N., Gummeson, J., Irwin, D.E., & Shenoy, P.J. (2010). Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems. *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 1-9. Article (accessed on 12 June 2023).

Sharma, N., Sharma, P., Irwin, D., & Shenoy, P. (2011, October). Predicting solar generation from weather forecasts using machine learning. *2011 IEEE international conference on smart grid communications (SmartGridComm)* (pp. 528-533). IEEE. Article (accessed on 12 Jan. 2023).

Sözen, A.; Arcaklıoğlu, E.; Özalp, M.; Çağlar, N. Forecasting based on neural network approach of solar potential in Turkey. *Renew.*

Energy 2005, *30*, 1075–1090. Article [CrossRef] (accessed on 12 June 2023).

Sözen, A.; Arcaklioğlu, E.; Özalp, M.; Kanit, E.G. Use of artificial neural networks for mapping of solar potential in Turkey. *Appl. Energy 2004*, *77*, 273–286. Available online: Article. [CrossRef] (accessed on 12 June 2023)

Tetko IV, Karpov P, Van Deursen R et al (2020) State-of-the-art augmented NLP Transformer Models for Direct and Single-step Retrosynthesis. *Nature Commun* 11(1):1–11. Article Google Scholar (accessed on 12 Jan. 2023).

Theocharides, S.; Makrides, G.; Livera, A.; Theristis, M.; Kaimakis, P.; Georghiou, G.E. Day-ahead photovoltaic power production forecasting methodology based on machine learning and statistical post-processing. Appl. . *Energy 2020*, *268*, 115023. Article (accessed on 12 June 2023).

Using AI and Weather Forecast To Optimize Renewable Energy Output Article (accessed on 12 Jan. 2023).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. Article .Google Scholar (accessed on 12 Jan. 2023).

Wang ,F., Xuan, Z., Zhen ,Z., Li, Y., Li ,K., Zhao ,L., Shafie-khah M., & Catala˜o ,J. P.,(2020). A minutely solar irradiance forecasting method based on real-time sky image-irradiance mapping model, *Energy Conversion and Management*, 220, 113075,. Article (accessed on 12 Jan. 2023).

Wang, F., Yu, Y., Zhang, Z., Li, J., Zhen, Z., & Li, K. (2018). Wavelet decomposition and convolutional LSTM networks based improved deep learning model for solar irradiance forecasting. *applied sciences*, *8*(8), 1286. Article (accessed on 12 Jan. 2023).

Wang, F.; Mi, Z.; Su, S.; Zhao, H. Short-term solar irradiance forecasting model based on artificial neural network using statistical feature parameters. *Energies 2012*, 5, 1355–1370. Article (accessed on 12 June 2023).

Wang, F.; Yu, Y.; Zhang, Z.; Li, J.; Zhen, Z.; Li, K. Wavelet decomposition and convolutional LSTM networks based improved deep learning model for solar irradiance forecasting. *Appl. Sci*. 2018, 8, 1286 Available online: https://www.mdpi.com/2076-3417/8/8/1286. Article (accessed on 12 June 2023).

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J. and Sun, L. (2022), Transformers in Time Series: A Survey, *arXiv preprint*, arXiv:2202.07125. (accessed on 12 June 2023).

Wu, N., Green, B., Ben, X., & O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*. Article Google Scholar (accessed on 12 Jan. 2023).

Zafar, R.; Vu, B.H.; Husein, M.; Chung, I.Y. Day–Ahead Solar Irradiance Forecasting Using Hybrid Recurrent Neural Network with Weather Classification for Power System Scheduling. Appl. Sci. 2021, 11, 6738 Available online: https://www.mdpi.com/2076-3417/11/15/6738 . Article (accessed on 12 June 2023).

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 35(12), 11106-11115. https://doi.org/10.1609/aaai.v35i12.17325 (accessed on 12 June 2023).

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021, May). Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 12, pp. 11106-11115).Google Scholar (accessed on 12 Jan. 2023).

Appendix A: source codes

```
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
        Normalizer
from sklearn.model_selection import train_test_split
from collections import Counter
from scipy import stats
import tensorflow as tf
import sklearn.preprocessing
from sklearn.metrics import r2_score
from keras.layers import Dense,Dropout,SimpleRNN,LSTM
from keras.models import Sequential
import pytz # timezones

from sklearn.linear_model import LinearRegression # Linear regression
from sklearn.ensemble import RandomForestRegressor # random forest
        regression
from sklearn.neural_network import MLPRegressor # neural network
        regression
from sklearn.svm import SVR # support vector regression



!pip install net

data = pd.read_csv("/content/drive/MyDrive/NEU/Transformer/Solar-
        Irradiance-Forecasting-using-ANNs-from-Scratch-main/Solar-
        Irradiance-Forecasting-using-ANNs-from-Scratch-
        main/SolarPrediction.csv")

data.head()


data.info()

data.describe()


data.head()

Feature Engineering
```

First step upon importing the dataset was to convert time and date parameters into a more useful format and add some coloumns that may be useful for visualisation, modelling and analysis.

```
def ingest_data(SolarPrediction):
  '''Read data from a CSV file and construct a pandas DataFrame
  Inputs:
    filename as string
  Outputs:
    df as DataFrame
  '''
  # read csv file
  data = pd.read_csv(SolarPrediction)
  data['Hour'] = pd.to_datetime(data['Time']).dt.hour
  # 'Data' column is unused. All elements contain the same value.
  # 'Time' is redundant and superseded by UNIXTime.
  data.drop(['Data','Time'],axis=1,inplace=True)

  # interpret columns as appropriate data types to ensure compatibility
  data['UNIXTime']      = pd.to_datetime(data['UNIXTime'],unit='s')
  data['Radiation']     = data['Radiation'].astype(float)
  data['Temperature']   = data['Temperature'].astype(float) # or int
  data['Pressure']      = data['Pressure'].astype(float)
  data['Humidity']      = data['Humidity'].astype(int) # or int
  data['WindDirection(Degrees)'] =
        data['WindDirection(Degrees)'].astype(float)
  data['Speed']         = data['Speed'].astype(float)
  data['TimeSunRise']   =
        pd.to_datetime(data['TimeSunRise'],format='%H:%M:%S')
  data['TimeSunSet']    =
        pd.to_datetime(data['TimeSunSet'],format='%H:%M:%S')
  data.rename(columns={'WindDirection(Degrees)': 'WindDirection',
        'Speed': 'WindSpeed'}, inplace=True)

  # compute length of each day
  data['DayLength'] = (data['TimeSunSet']-
        data['TimeSunRise'])/np.timedelta64(1, 's')

  # we don't need sunrise or sunset times anymore, so drop them
  data.drop(['TimeSunRise','TimeSunSet'],axis=1,inplace=True)

  # index by UNIX time
  data.sort_values('UNIXTime', inplace=True) # sort by UNIXTime
  data.set_index('UNIXTime',inplace=True) # index by UNIXTime

  # Localize the index (using tz_localize) to UTC (to make the
        Timestamps timezone-aware) and then convert to Eastern (using
        tz_convert)
  hawaii=pytz.timezone('Pacific/Honolulu')
  data.index=data.index.tz_localize(pytz.utc).tz_convert(hawaii)
```

```
    # assign unit labels to data keys
    units={'Radiation':'W/m^2','Temperature':'F','Pressure':'in
        Hg','Humidity':'\%','DayLength':'sec'}
    return data, units

from datetime import datetime
from pytz import timezone
import pytz
hawaii= timezone('Pacific/Honolulu')
data.index =  pd.to_datetime(data['UNIXTime'], unit='s')
data.index = data.index.tz_localize(pytz.utc).tz_convert(hawaii)
data['MonthOfYear'] = data.index.strftime('%m').astype(int)
data['DayOfYear'] = data.index.strftime('%j').astype(int)
data['WeekOfYear'] = data.index.strftime('%U').astype(int)
data['TimeOfDay(h)'] = data.index.hour
data['TimeOfDay(m)'] = data.index.hour*60 + data.index.minute
data['TimeOfDay(s)'] = data.index.hour*60*60 + data.index.minute*60 +
        data.index.second
data['TimeSunRise'] = pd.to_datetime(data['TimeSunRise'],
        format='%H:%M:%S')
data['TimeSunSet'] = pd.to_datetime(data['TimeSunSet'],
        format='%H:%M:%S')
data['DayLength(s)'] = data['TimeSunSet'].dt.hour*60*60 \
                    + data['TimeSunSet'].dt.minute*60 \
                    + data['TimeSunSet'].dt.second \
                    - data['TimeSunRise'].dt.hour*60*60 \
                    - data['TimeSunRise'].dt.minute*60 \
                    - data['TimeSunRise'].dt.second
data.drop(['Data','Time','TimeSunRise','TimeSunSet'], inplace=True,
        axis=1)
data.head()
```

Feature Visualisation

Next, in order to get a better understanding of the data, hourly and monthly means of several variables were visualised using bar plots.

```
data, units =
        ingest_data('/content/drive/MyDrive/NEU/Transformer/Solar-
        Irradiance-Forecasting-using-ANNs-from-Scratch-main/Solar-
        Irradiance-Forecasting-using-ANNs-from-Scratch-
        main/SolarPrediction.csv')
print(data.head())

sns.set(style="white")

# make IPython render plots inline
%matplotlib inline
```

Plotting libraries are imported to visualize data. Then each measurement is visualized and Pearson correlations are calculated to determine which parameters have the most impact on one another.
First, a basic correlation matrix is generated to weed out irrelevant data and identify the most significant features in the set.

```
def corrPairs(data):
   '''Pairwise correlation matrix'''
   corr = data.corr() # Compute the correlation matrix
   mask = np.zeros_like(corr, dtype=np.bool) # make mask
   mask[np.triu_indices_from(mask)] = True # mask upper triangle
   sns.heatmap(corr, mask=mask, cmap='coolwarm', center=0,
         square=True, linewidths=.5, annot=True, cbar=False)

data['WeekOfYear'] = data.index.week # add week to view correlation
plt.figure(figsize=(6,6))
corrPairs(data)


sns.heatmap(data.corr(),cmap="crest")
plt.show()

def corrfunc(x, y, **kws):
   '''add pearson r correlation to plots'''
   r, _ = stats.pearsonr(x, y)
   ax = plt.gca()
   ax.annotate("r = {:.2f}".format(r),xy=(.1, .9), xycoords=ax.transAxes,
         color='white')
   return

def corrMap(data,features):
   '''plot bivariate correlations'''
   g = sns.PairGrid(data, vars=features)
   g.map_upper(plt.scatter, s=10)
   g.map_diag(sns.distplot, kde=False)
   g.map_lower(sns.kdeplot, cmap="coolwarm", shade=True,
         n_levels=30)
   g.map_lower(corrfunc)
   g.map_lower(corrfunc)


feats = { 'Temperature':'red', 'Humidity':'green', 'Pressure': 'blue' }
for i in feats:
   count = Counter(data[i])
   plt.bar(count.keys(), count.values(), color=feats[i])
   plt.title('Distribution')
   plt.ylabel('Occurrence')
```

```
    plt.xlabel(i)
    plt.show()

#Plot solar radiation against temperature
plt.figure(figsize=(24,8))
sns.barplot(x=data['Temperature'].round(decimals=0),y=data['Radiation'])
plt.xlabel('Temperature (C)')
plt.ylabel('Solar Radiation (kW/h)')
plt.title('Solar-Irradiance versus Temperature')
plt.show()

#Take hourly mean of the dataset. Plot solar radiation against hours in a
        day.
rad_vs_hour= data.loc[:, ['Radiation', 'Hour']].groupby('Hour').mean()
rad_vs_hour.plot(kind='bar')
plt.xlabel('Time of the day (hour)')
plt.ylabel('Radiation(W/m2)')
plt.title('Total Radiation per hour of the day')
plt.show()

feature_list=['Radiation','Temperature','Humidity','Pressure']
# bivariate density matrix
corrMap(data,feature_list)
plt.show()

def color_y_axis(ax, color):
    '''Color y axis on two-axis plots'''
    for t in ax.get_yticklabels():
        t.set_color(color)
    ax.yaxis.label.set_color(color)
    return None

def plotVs(data,timescale,feature1,feature2,ax1,units):
    '''Plot feature vs radiation'''
    ax2=ax1.twinx()
    data_grouped= data.groupby(timescale)

    data_feature1 = data_grouped[feature1].mean()
    data_feature1_errorpos =  data_feature1+data_grouped[feature1].std()/2
    data_feature1_errorneg =  data_feature1-data_grouped[feature1].std()/2
    ax1.plot(data_feature1)
    ax1.fill_between(data_feature1.index, data_feature1_errorpos.values,
        data_feature1_errorneg.values, alpha=0.3, antialiased=True)
    ax1.set_ylabel(feature1+' '+units[feature1])
    color_y_axis(ax1, 'b')

    if feature2 == 'Radiation':
        rad = data_grouped['Radiation'].mean()
        ax2.plot(rad,'r')
```

```
    ax2.fill_between(data_feature1.index, 0, rad, alpha=0.3,
      antialiased=True, color='red')
    ax2.set_ylabel('Radiation'+' '+units['Radiation'])
    color_y_axis(ax2, 'r')
  else:
    data_feature2 = data_grouped[feature2].mean()
    data_feature2_errorpos =
      data_feature2+data_grouped[feature2].std()/2
    data_feature2_errorneg =  data_feature2-
      data_grouped[feature2].std()/2
    ax1.plot(data_feature2)
    ax1.fill_between(data_feature2.index, data_feature2_errorpos.values,
      data_feature2_errorneg.values, alpha=0.3, antialiased=True)
    ax1.set_ylabel(feature2+' '+units[feature2])
    color_y_axis(ax1, 'g')
  return ax1, ax2

def HourlyWeeklyVs(df,feature1,feature2,units):
  '''Plot a feature vs radiation for time of day and week of year'''
  plt.figure(figsize=(18, 6))
  ax=plt.subplot(121) # hourly
  ax1,ax2 = plotVs(data,data.index.hour,feature1,feature2,ax,units)
  lines1, labels1 = ax1.get_legend_handles_labels()
  lines2, labels2 = ax2.get_legend_handles_labels()
  ax2.legend(lines1 + lines2, labels1 + labels2)
  plt.xlabel('Hour of Day (Local Time)')
  plt.title('Mean Hourly {0} vs. Mean Hourly
      {1}'.format(feature1,feature2))

  ax=plt.subplot(122) # weekly
  ax1, ax2 = plotVs(data,pd.Grouper(freq='W'),feature1,feature2,ax,units)
  lines1, labels1 = ax1.get_legend_handles_labels()
  lines2, labels2 = ax2.get_legend_handles_labels()
  ax2.legend(lines1 + lines2, labels1 + labels2)
  plt.xlabel('Week of Year')
  plt.title('Mean Weekly {0} vs. Mean Weekly
      {1}'.format(feature1,feature2))
  return

for feature in feature_list[1:]: # radiation vs feature
  HourlyWeeklyVs(data,feature,feature_list[0],units)
plt.show()

data.drop(['WindDirection','WindSpeed'], axis=1, inplace=True) # drop
      irrelevant features
```

3. Training & Testing

We desire an algorithm that will predict values (radiation for a given set of inputs) There are many models to choose from, and more than one may be appropriate.

In this analysis, we will try several models and compare their performance to evaluate the best algorithm to predict solar radiation.

Linear Regression
Random Forest Regression
Neural Network Regression
Support Vector Regression

```
# IMPORT ML CLASSIFIERS
from sklearn.linear_model import LinearRegression # Linear regression
from sklearn.ensemble import RandomForestRegressor # random forest
    regression
from sklearn.neural_network import MLPRegressor # neural network
    regression
from sklearn.svm import SVR # support vector regression
```

3.1. Preparing the algorithm

Even before we downselect to a specific model, we can prepare a prediction algorithm that takes in our data and makes a prediction. Using scikit-learn, it is easy to swap out different models and maintain the same higher-level structure to the program.

To train the algorithm, we implement a split train/test methodology to prevent bias in the learning. The dataset is split into a randomly sampled pool of datapoints. 80% of those points are used for training, the remaining 20% is used for validation of the training data. So the test data is not necessarily continuous time, but rather a random selection of points from the set.

For demonstration purposes, we use the entire dataset (including training and test points) to visualize algorithm performance over time. This is inherently biased, since some of the points we will see will have been points that the algorithm has already trained on and potentially optimized to. However, we validate the algorithm accuracy against the subset of testing points (which the were not used for training), so we can still be confident in evaluating the performance using the accuracy metric and by keeping this potential bias in mind.

```
x = data.drop('Radiation',axis=1).to_numpy()
y = data['Radiation'].to_numpy()
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
        random_state=1)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)

from sklearn import preprocessing # ML tools
from sklearn.model_selection import train_test_split # split data

from bokeh.plotting import figure, show, output_notebook

def plot_test(clf,X_test,y_test):
   y_predicted = clf.predict(X_test)

   p = figure(tools='pan,box_zoom,reset',x_range=[0, 100], title='Model
        validation',y_axis_label='radiation')
   p.grid.minor_grid_line_color = '#eeeeee'

   p.line(range(len(y_test)),y_test,legend='actual',line_color='blue')

        p.line(range(len(y_test)),y_predicted,legend='prediction',line_color
        ='red')
   output_notebook()
   show(p)
   return

def plot_real(clf,x,y_actual,index):
   ''' Plot predictions for actual measurements.
   inputs:
      clf        as classifier   the trained algorithm
      x          as array        timeseries of measurement inputs
      y_actual   as array        corresponding timeseries of actual results
   '''
   y_predicted = clf.predict(x)

   p = figure(toolbar_location='right', title='Predicted vs
        Actual',y_axis_label='radiation',x_axis_type="datetime")
   p.grid.minor_grid_line_color = '#eeeeee'
   p.line(index,y_actual,legend='actual',line_color='blue')
   p.line(index,y_predicted,legend='prediction',line_color='red')
   output_notebook()
   show(p)
   return

def train_model(X,y,clf,debug=False):
   ''' Train algorithm.
   inputs:
      X      as array      features
```

```
        y       as array       label(s)
        clf     as scikit-learn classifier (untrained)
    returns:
        clf     as trained classifier
        accuracy  as float
    '''
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
    model = clf.fit(X_train,y_train)
    accuracy = clf.score(X_test,y_test)
    return clf, model, accuracy, X_test, y_test

def go(x,y,algorithm,debug=True):
    ''' Easy model train and test. '''
    clf, model, accuracy, X_test,
            y_test=train_model(x,y,algorithm,debug=True)
    print('Accuracy: %s percent'%str(accuracy*100))

    if debug:
        plot_test(clf,X_test,y_test)
        plot_real(clf,x,y,data.index.values)
    return

from sklearn import preprocessing # ML tools
from sklearn.model_selection import train_test_split # split data
from sklearn.metrics import accuracy_score, f1_score, precision_score,
            recall_score, mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression

from bokeh.plotting import figure, show, output_notebook

def plot_test(clf,X_test,y_test):
    y_predicted = clf.predict(X_test)

    p = figure(tools='pan,box_zoom,reset',x_range=[0, 100], title='Model
            validation',y_axis_label='radiation')
    p.grid.minor_grid_line_color = '#eeeeee'

    p.line(range(len(y_test)),y_test,legend='actual',line_color='blue')

            p.line(range(len(y_test)),y_predicted,legend='prediction',line_color
            ='red')
    output_notebook()
    show(p)
    return

def plot_real(clf,x,y_actual,index):
    ''' Plot predictions for actual measurements.
    inputs:
        clf        as regressor   the trained algorithm
        x          as array       timeseries of measurement inputs
```

```
        y_actual    as array       corresponding timeseries of actual results
    '''
    y_predicted = clf.predict(x)

    p = figure(toolbar_location='right', title='Predicted vs
          Actual',y_axis_label='radiation',x_axis_type="datetime")
    p.grid.minor_grid_line_color = '#eeeeee'
    p.line(index,y_actual,legend='actual',line_color='blue')
    p.line(index,y_predicted,legend='prediction',line_color='red')
    output_notebook()
    show(p)
    return

def train_model(X,y,clf,debug=False):
    ''' Train algorithm.
    inputs:
        X       as array       features
        y       as array       label(s)
        clf     as scikit-learn regressor (untrained)
    returns:
        clf     as trained regressor
        metrics as dict        regression metrics (MSE, RMSE, MAE, R^2)
        X_test  as array       test features
        y_test  as array       test labels
    '''
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
    model = clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    metrics = {'MSE':mse, 'RMSE':rmse, 'MAE':mae, 'R^2':r2}

    accuracy = None
    if debug:
        accuracy = r2
        plot_test(clf,X_test,y_test)

    return clf, metrics, X_test, y_test

def go(x,y,algorithm,debug=True):
    ''' Easy model train and test. '''
    clf, metrics, X_test, y_test=train_model(x,y,algorithm,debug=True)
    print('Metrics: ', metrics)

    if debug:
        plot_real(clf,x,y,data.index.values)
```

    return clf, metrics, X_test, y_test

3.2. Linear Regression

Let's implement the first ML algorithm: Linear regression.

Linear regression is probably the simplest fit, but weather characteristics are probably quite nonlinear. Regardless, let's see how it performs -- it might be good enough

go(x,y,LinearRegression())

3.3. Random Forest Regression

Another algorithm to try is random forest regression. This works in a fundamentally different way to linear regression, so maybe we'll have more success. Most importantly, this algorithm can handle nonlinear inputs.

go(x,y,RandomForestRegressor())

3.4. Neural Network Regression
Neural Networks are very tunable to suit a wide variety of problems. In this case, a neural network will be used to optimize squared error. Since this is just an exploration, we use default parameters knowing that performance may be much different if these values are tuned to suit our problem

go(x,y,MLPRegressor())

Wow, worse than linear regression! Although better results are probably possible with this algorithm, we already have random forest regression performing north of 90% accuracy. Tuning the neural network is not really worth the trouble at this point.

3.5. Support Vector Regression
This is another algorithm that comes packaged with scikit-learn. Let's implement it without digging into the theory, just to see how it performs out of the box.

go(x,y,SVR())

3.6 Gradient Boosting Regressor:

from sklearn.ensemble import GradientBoostingRegressor
go(x, y, GradientBoostingRegressor())

3.7 K-Nearest Neighbors Regressor:

```
from sklearn.neighbors import KNeighborsRegressor
go(x, y, KNeighborsRegressor())
```

3.8 Decision Tree Regressor:

```
from sklearn.tree import DecisionTreeRegressor
go(x, y, DecisionTreeRegressor())
```

3.9 Ridge Regression:

```
from sklearn.linear_model import Ridge
go(x, y, Ridge())
```

3.10 Lasso Regression:

```
from sklearn.linear_model import Lasso
go(x, y, Lasso())
```

3.11 ElasticNet Regression:

```
from sklearn.linear_model import ElasticNet
go(x, y, ElasticNet())
```

To adjust the hyper parameters of each model as needed to achieve optimal performance and also, to make sure to use the evaluation metrics provided in the code to assess the performance of each model let us use Artificial Neural Network as an example

```
def train_model(x, y, clf, debug=False):
   ''' Train a model, output accuracy '''
   X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
      random_state=42)
   clf.fit(X_train, y_train)
   model = clf
   y_pred = clf.predict(X_test)
   mse = mean_squared_error(y_test, y_pred)
   rmse = np.sqrt(mse)
   mae = mean_absolute_error(y_test, y_pred)
   r2 = r2_score(y_test, y_pred)
   if debug:
      print(f"MSE: {mse:.4f}\nRMSE: {rmse:.4f}\nMAE: {mae:.4f}\nR2
      Score: {r2:.4f}")
   return clf, model, mse, rmse, mae, r2, X_test, y_test, y_pred
```

```
def go(x, y, algorithm, debug=True, **kwargs):
    ''' Easy model train and test. '''
    clf = algorithm(**kwargs)
    clf, model, mse, rmse, mae, r2, X_test, y_test, y_pred = train_model(x,
        y, clf, debug=True)
    return clf, model, mse, rmse, mae, r2, X_test, y_test, y_pred



go(x, y, MLPRegressor, activation='identity', learning_rate_init=0.001)


from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error,
    r2_score

def train_model(X, y, clf, debug=False):
    ''' Train algorithm.
    inputs:
        X       as array        features
        y       as array        label(s)
        clf     as scikit-learn regressor (untrained)
    returns:
        clf     as trained regressor
        accuracy  as float
    '''
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    model = clf.fit(X_train, y_train)
    y_predicted = clf.predict(X_test)

    mse = mean_squared_error(y_test, y_predicted)
    rmse = mean_squared_error(y_test, y_predicted, squared=False)
    mae = mean_absolute_error(y_test, y_predicted)
    r2 = r2_score(y_test, y_predicted)

    print("MSE: ", mse)
    print("RMSE: ", rmse)
    print("MAE: ", mae)
    print("R2: ", r2)

    if debug:
        plot_test(clf, X_test, y_test)
        plot_real(clf, X, y, data.index.values)

    return clf, model, mse, rmse, mae, r2, X_test, y_test

def go(x, y, algorithm, debug=True, **kwargs):
    ''' Easy model train and test. '''
```

```
    clf = algorithm(**kwargs)
    clf, model, mse, rmse, mae, r2, X_test, y_test = train_model(x, y, clf,
        debug=True)

    if debug:
        print('MSE: %s'%str(mse))
        print('RMSE: %s'%str(rmse))
        print('MAE: %s'%str(mae))
        print('R2: %s'%str(r2))

    return clf, model, mse, rmse, mae, r2, X_test, y_test



go(x, y, RandomForestRegressor, n_estimators=100, max_depth=5)
```

To recap, recall the accuracy of each algorithm attempted so far:

Linear Regression: ~60%
Random Forest Regression: >90%
Neural Network Regression: ~50%
Support Vector Regression: <50%
Thus we select Random Forest Regression as our algorithm for turning

4. Tuning the Algorithm
Now let's consider how we can improve the accuracy of our model.

Here's what the scikit-learn documentation say:

In random forests (see RandomForestClassifier and RandomForestRegressor classes), each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

On a high level, regression derived from decision trees often results in low bias, high variance models, and is prone to overfitting. While the random forest method (which is built upon many decision trees) is more robust against bias and variance, overfitting is still a potential pitfall.

For random forests, there are three main tuning parameters:

Number of trees. (n_estimators) More is better, with diminishing returns. Obviously more trees means longer compute times. A critical number of trees must be found where significant accuracy and compute times are optimized.

Number of features to consider at each split. (max_features) If some trees consider a different subset of features than others, the correlation between those two groups is minimal. This is desirable because it teases out the influence of each individual feature.

Depth of trees. (max_depth) Having trees go too deep can lead to overfitting. There is a critical depth where the trees split enough to result in useful fit without being too influenced by single values. Depth may instead be constrained by min_samples_split, min_samples_leaf, min_weight_fraction_leaf, or max_leaf_nodes rather than specifying tree depth outright.

```
# DEFAULT VALUES
  RandomForestRegressor(n_estimators=10,
                criterion='mse',
                max_depth=None,
                min_samples_split=2,
                min_samples_leaf=1,
                min_weight_fraction_leaf=0.0,
                max_features='auto',
                max_leaf_nodes=None,
                min_impurity_decrease=0.0,
                min_impurity_split=None,
                bootstrap=True,
                oob_score=False,
                n_jobs=1,
                random_state=None,
                verbose=0,
                warm_start=False)
```

Start by seeing if performance improves by simply increasing the number of trees.

```
# default algorithm for reference
print('Default random forest regressor:')
go(x,y,RandomForestRegressor,debug=False)

# tuning round 1
print('Tuned regressor:')
go(x,y,RandomForestRegressor(n_estimators=100, n_jobs=-
    1),debug=False)
```

Appendix B: Running Transformer codes

Solar Irradiance Transformer code and further description can be downloaded at:
https://github.com/kayodeakanni/
and
https://drive.google.com/drive/folders/1JhlOjD5bh7S61NB7afAXsWzwsfvGwhYf

Appendix C: Similarity Report

check 0

ORIGINALITY REPORT

14%	11%	8%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

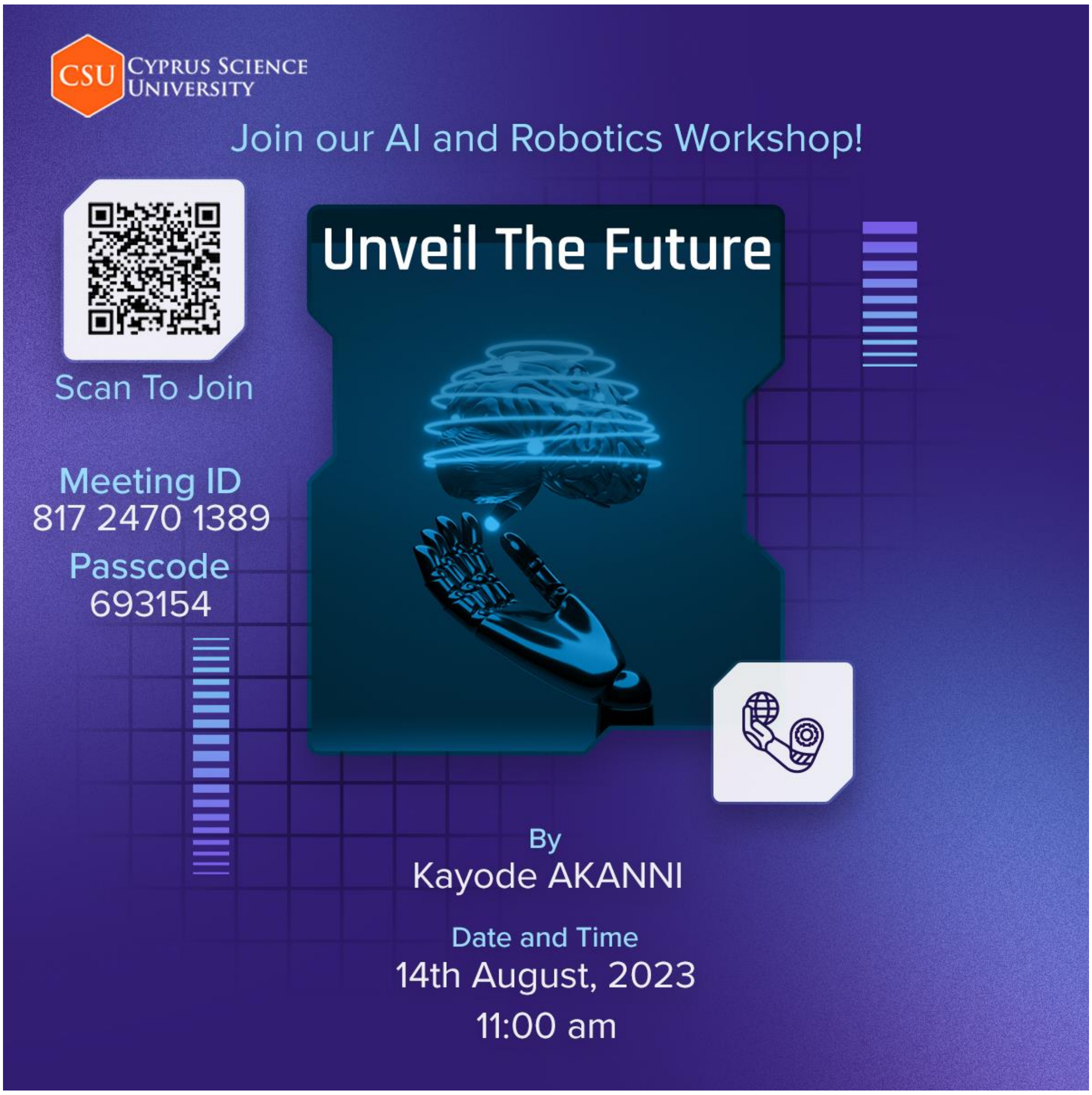
1	docs.neu.edu.tr Internet Source	2%
2	www.mdpi.com Internet Source	2%
3	dokumen.pub Internet Source	1%
4	www.kaggle.com Internet Source	<1%
5	www.researchgate.net Internet Source	<1%
6	digitalcommons.dartmouth.edu Internet Source	<1%
7	Yogendra Narayan Pandey, Ayush Rastogi, Sribharath Kainkaryam, Srimoyee Bhattacharya, Luigi Saputelli. "Machine Learning in the Oil and Gas Industry", Springer Science and Business Media LLC, 2020 Publication	<1%
8	diglib.tugraz.at Internet Source	<1%
9	huggingface.co Internet Source	<1%

Appendix D: Publications based on the Thesis and invitations

Publications

1- Application of Transformers in Information Security: Current Trends and Prospects
https://ieeexplore.ieee.org/document/10102203
2- BERT-IDS: An Intrusion Detection System Based On Bidirectional Encoder Representations from Transformers. In print
3- Reviewing Applications Of Artificial Intelligence And Blockchain In Energy Industry. In print
4 -Transformer model for Solar Irradiance Forecasting: Optimizing energy case. In print

Invitations



11 August 2023

ITU Member States,
Sector Members,
Associates and Academia,
Heads of UN Agencies,
UN Missions in New York,
Broadband Commissioners,
SDG Digital Advisory Group,
SDG Digital Supporters,
P2C Pledgers

Contact: sdgdigital@itu.int

Subject: SDG Digital, 17 September 2023, New York

Dear Sir/Madam,

We are pleased to invite your organization to **SDG Digital**, taking place on **Sunday, 17 September 2023** from **10:15 to 16:30 EST**, in the ECOSOC Chamber at United Nations Headquarters in New York.

SDG Digital is a collaborative effort by the International Telecommunication Union (ITU) and the United Nations Development Programme (UNDP), with the support of the UN System.

This event is organised in conjunction with the 2023 SDG Summit on 18-19 September, hosted by the United Nations General Assembly, as we mark the mid-point on the road to 2030. The UN Secretary-General will convene an SDG Action Weekend from 16-17 September 2023, focused on showcasing what is possible if the global community comes together, across countries, sectors and systems, to bring solutions for the Sustainable Development Goals to scale – including through launch of a set of High Impact Initiatives.

During SDG Digital, we will hear from leaders from government, civil society, technology executives, think tanks and academia, and catalyze concrete opportunities for engagement around issues of inclusive and responsible digital transformation for implementation of the 2030 Agenda.

SDG Digital

Highlights of SDG Digital will include:

- Unveiling of the **SDG Digital Acceleration Plan**, supported by the Boston Consulting Group (BCG) as our SDG Digital knowledge partner – featuring 34 inspiring digital solutions and 8 country success stories – which outlines actionable steps with appropriate safeguards and collaborative efforts to bridge the gaps needed to deliver the 2030 Agenda.
- Announcement of the winners from the **SDG Digital Gamechangers** competition, which honors individuals and organizations working towards rescuing the Global Goals through digital.
- Announcement of new pledges and commitments for universal meaningful connectivity under the umbrella of the Partner2Connect Digital Coalition.
- Launch of the UN High Impact Initiative on **Digital Public Infrastructure**.

We would be honoured if you could join us in New York on 17 September. SDG Digital is designed to inspire and catalyse action around the use of digital technologies to accelerate SDG achievement between now and 2030.

We invite you to visit this webpage where you can find further information about the event, and the link to register. Do not hesitate to contact the organizing team at sdgdigital@itu.int for additional support. Please submit your registration request as early as possible, but no later than **4 September 2023** as space is limited. Please also visit the SDG Digital website for periodic updates on the programme and speakers.

We look forward to welcoming you to this unique and exciting event as your contribution and insights would greatly enrich SDG Digital.

Please be assured of our highest consideration.

Yours faithfully,

Doreen Bogdan-Martin Secretary-General International Telecommunication Union	Achim Steiner Administrator United Nations Development Programme

Appendix E: CV

Personal information 	**Olukayode Akanni (ASME, COREN, NSE, IASSC certified Black Belt Six Sigma, Black in AI, Black in Robotics, OpenMined, WAIE, Intel Edge AI, MD4SG, Meetups NeurIPS) IT/Program Manager, AI/Software Engineer** **Contact Address**: Apt. 3, Kavaz E building, Piabella (Kayaz Harbour Sitesi),Mersin 10, Cyprus. **Telephone Number:** +234-8035341567, +90 5488504054 **E-mail**: kayodeakanni@gmail.com Europas CV link: https://resume.io/r/GNjVxuXSl LinkedIn https://ng.linkedin.com/in/kayode-akanni-cssbb-mba-ai-66789626 Personal website https://about.me/kayodeakanni Github Portfolio https://github.com/kayodeakanni Google Scholars https://scholar.google.com/citations?user=MnNSeJIAAAAJ&hl=en&citsig=AMD79oqUUyT89KzpeO1tbzJhpEuQ7egg0A NetRights Coalition https://cpj.org/wp-content/uploads/2020/09/PIN-Memo-on-draft-DPB.docx.pdf Action Coalition on Civic Engagement in AI Design: https://ecnl.org/focus-areas/technology-and-artificial-intelligence
Educational Qualification	(June 2023) Masters of Science, Artificial Intelligence Engineering, AI Engineering Dept., Research Center for AI and IoT, AI and Robotics Institute, Near East University, Mersin 10, Turkey. (2013) Technology Entrepreneurship, Finance, Venture lab, Management Science and Engineering Stanford University, United States of America (2012- 2012) MSC, Operation Research, Business Administration Department, UNILAG, Lagos(uncompleted) (2007-2010) NOUN, Master in Business Administration-MBA (Information Technology) (1995-2001) University of Ibadan, Ibadan, B.Sc. Hons Mechanical Engineering, Second Class Lower Division (1988- 1994) Abadina College, University of Ibadan, Ibadan. (S. S. C.E) May/June1994.-6 Distinctions and 3 Credits. (1984-1987) Polytechnic Staff School, The Polytechnic, Sango, Ibadan (1982-1983)Trinity Nursery and Primary School, Ojoo, Ibadan, First School Leaving Cert.
Summary	• I am a professional working on digital inclusion/ digital rights of citizen and also love applying / building AI apps. My latest AI web app is AFri News Multilingual Embedding. This app leverages multilingual semantic model from COhere.ai to revolutionize media and news industry for multilingual market like Africa by Enabling any person to track news in real time without translating or understanding other regional languages. Right now, I am working in a team on AIOT Health Mobile App, which processes the AI algorithms locally on a hardware device to mobile platform. I have worked on virtual salon solution, a face swap AI web app for Nail and hair salon at Velena.com, AI and cybersecurity applications, AI and energy applications, AI and attendance system IOT applications. Drones, Raspberry Pi, Arduino and Rights Respecting AI framework
Objective	To obtain a good position that provides opportunity for rewarding career and using Engineering, business, AI and robotics- as a key tool for sustainable development. I have proven ability to take ownership and deliver excellent results with attention to details.
Previous work Experience	2022-2023: Software Engineer, **Research Centre for AI and IoT, AI and Robotics Institute, Near East University, Mersin 10, Turkey** ● Built in a team, a full stack e-commerce application using PHP/JavaScript/HTML/Bootstrap https://velenasalon.com ● Participated in IOT Projects of the institute.

	● Participated in a team in the design and development of an Artificial Intelligence enabled mobile and web health app. 2022: **ECNL's Action Coalition Partner, European Center for Not-For-Profit Law Stichting, Netherlands | Knowledge House (KnowledgeHouseAfrica-KHA),** Girne KKTC, Mersin,10, Turkey. ●Ensure the growth and improvements in the works of DesignIT International rebranded as Knowledge House with presence in Europe as a parent social enterprise for her activities in Europe and Africa. ● Execute our Action Coalition on Civic Engagement in AI Design. This is an initiative launched under the auspices of the "Tech for Democracy Initiative" spearheaded by the Ministry for Foreign Affairs and Ministry for Development Cooperation of Denmark. and we have been working on developing a Guide for meaningful Engagement and participation of CSOs/affected communities in the development of human rights impact assessment (HRIAs) of AI-driven systems and have created a draft framework for meaningful trustworthy engagement with domain experts and stakeholders across law enforcement, government, NGOs and the private sectors, while using Privacy- preserving AI exposure in Openmined to also develop socio-technical approaches and frameworks to enable privacy, security and trust in the data sharing and AI applications. ● Consider issues of privacy, security and trust as they relate to data sharing and curation of sensitive data sets. 2012- 2021: **DesignIT International aka KnowledgeHouseAfrica,** #27, Josade way, Agunfoye-Adamo Rd, Adamo. Ikorodu. Lagos, Nigeria. - Leading the growth and impact of a nonprofit social enterprise organization dedicated to apply ICT, builds an ICT-enabled support systems and things, digital inclusion, advocates digital rights and AI – related legislation in order to improve livelihoods for underserved, unconnected and the unborn using Free and Open Source Technologies as a key tool. - Serves as Black in robotics Teaching assistance in Robotic Education outreaches including from Imagination to Reality: Computer-Aided Design using Auto desk Tinkercad, in partnership with Robomechanics lab at Carnegie Mellon University, CMU, USA. - Conduct research in development and application of privacy and security related research including data anonymization and synthetic data generation, differential privacy, federated learning, and other related technologies through Openmined, Black in AI, Black in Robotics, OpenMined, WAIE, Intel Edge AI, MD4SG, Meetups NeurIPS - Robotics Educator/Engineer using CAD program, design and using 3D printing bring our own creation to life: Describe the product development process, express product design ideas using 2D sketches, model a component with complex shapes, model an assembly of components with kinematic linkages, render and animate the appearance and functionality of a product, receive a 3D print of a product designed ourselves. 1997- 2001; **Research (Design) Assistant**, **Mechanical Engineering Dept., University of Ibadan, Ibadan.** - From Imagination to Reality: Computer-Aided Design. Developed engineering drawings of all machines in a plant-consultancy work with Oyo State Government & Raw Materials Development Research council (RMDRC);Planned, managed major engineering operations & supervised its production; Design, development, Installation & commissioning of 20 Tons capacity Oragno-mineral Fertilizer Plant, Ibadan; Developed an Environmental Information System for scheduling & forecasting of "**waste to wealth**" project.

Skills	***ICT Skills:*** AutoCAD, 3D CAD, Graphic Designs, MS & Primavera Project Planner. **Digital Skills** Desktop & Web Applications & technologies, Digital marketing, ICT4D Consultant, Programming (Python, Scratch, Arduino, JavaScript, CSS, HTML, MySQL, PhP, C++, Open source projects like Open CV, OpenVINO, ONNX, Tensor flow, Pytorch, FastAI. Experience in using S/w Development tools), AI/ML/DS **Soft Skills**: - Team leadership; Good oral and written communication; Excellent analytical skills; good organizational and interpersonal skills; Project management; contract management, designing and meeting budget and KPIs, CAPEX and, OPEX; Monitoring, evaluation & Impact assessment. Author, Speaker, Youth Coach.
Professional Membership	Member of Institution of Mechanical Engineers in view **(ImechE #8001124)** and **COREN** (**R30,180**), **IASSC Black Belt** (**#GR7640001970A), IRCA QMS** Auditor**(#SGSD/SSCE/QMSLAC/511493/P/26828**), **AI Saturdays Lagos (DL # C05/2/6272020/006**) Member of Nigerian Society of Engineers **(NSE-#21579), Nigeria** and Member of American Society of Mechanical Engineers **(ASME- #9355165). AI membership**: DataScienceNigeria, AISaturdays Lagos (, Tensorflow Lagos and GDG Lagos, and GDG Ikorodu, Black in AI, OpenMined, Data Native unlimited, Codeclub of Raspberry pi foundation. Internet Governance Forum (IGF)
Publications	1- Development and construction of cashew juice extractor machine (a Project work submitted in partial fulfillment of OND in Mechanical Engineering. 2- Design of 10kN capacity Screw jack (a Design project term paper). 3- Computer graphical representation of Organo-Mineral Fertilizer Pilot Plant Process Flow chart. (a project work submitted in partial fulfillment of B Sc. Hons in Mechanical Engineering). 4- West Africa Telecommunications and Nigeria (a report given to YIELD after WAfritel 2002), 5- E-readiness of Banks and Financial institutions in West Africa (a report given to YIELD after Finance IT Africa) 6- Internet and You (Presented at the National Youth Service Corps Camp, NCCF, Lagos State ,2002) 7- NYSC Service Year CD Rom: Nigeria Christian Corpers' Fellowship, Lagos State (2002 and 2003) 8- NGO and e-commerce: (Presented at the Development Information Network meeting, Lagos State (2003) 9- The Nigerian Youths Designing Open Source for Livelihood Opportunities: (Presented at the First African Conference on Digital Commons, South Africa, 2004) 10- What young people are doing @ WSIS. (Presented at the Information Communication and Technologies Youth Empowerment Conference 2004.) 11- Publication of United Nations Economic Commission for Africa on "African youth Speaks". (2004) 12-Global Process, Local Reality: Nigerian youth Lead Action in the Information Society.WSIS Policy II, Tunisia 2005 https://scholar.google.com/citations?user=MnNSeJIAAAAJ&hl=en&citsig=AMD79oqUUyT89KzpeO1tbzJhpEuQ7egg0A 13- Solving poverty through Digital Economy. AI6Lagos Data science & Machine learning Project. https://GitHub.com/deep-forthinkn , https://twitter.com/AISaturdayLagos/status/1208388686204809216?s=09(2019) 14- American Sign Language Translator, an Intel Edge AI Udacity's Winning AI for social good Project (2020) https://GitHub.com/ASL 15- Facial Expression Recognition, AI6 Lagos Deep learning project (2020), https://GitHub.com/AI6DLProject 16- OpenMined's privacy preserving ML with python tutorial pidgin translation. https://github.com/OpenMined/PySyft 17- AIOT Health App 18- Application of Transformers in Information Security: Current Trends and Prospects https://ieeexplore.ieee.org/document/10102203 19- BERT-IDS: An Intrusion Detection System Based On Bidirectional Encoder Representations from Transformers 20- Attendance System via Internet of Things, Blockchain and Artificial Intelligence Technology: A Systematic Literature Review

	https://www.researchgate.net/publication/369242900_Attendance_System_via_Internet_of_Things_Blockchain_and_Artificial_Intelligence_Technology_Literature_Review 21- Reviewing Applications Of Artificial Intelligence And Blockchain In Energy Industry 22 - 0ptimizing energy in the digital age: solar irradiance forecasting using transformer model
Community Activities	**Taking IT Global (TIG)**- An international organization-TIG brings together young people in more than 190 countries within international networks to collaborate on concrete projects addressing global problems and creating positive change **www.takingitglobal.org** **Free Software and Open Source Foundation for Africa (www.fossfa.net)**–An African based organisation made up of Open Source Users & Developers throughout Africa, devoted to the development & promotion of Free and Open Source Software in Africa **World Summit On Information Society Youth Africa (www.wsisyouth.org)** –The concerted effort of the African youth involvement in the WSIS Process, which is the initiative of the United Nations and the International Telecommunication Union (ITU) **African Youth and the Information Society Initiative** (**UNECA**)- Organized by the United Nations Economic Commission for Africa. A platform for African youth to share experiences and knowledge with stakeholders in order to help develop innovative approaches to their needs and to implement the UN World Summit on the Information Society action plan at country & regional level.
Hobbies	Thinking, planning, reading, writing, creative reasoning, leadership, decision making and problem solving.
Languages	English, Yoruba and French language
Awards	• **Intel Edge AI Udacity's Winning AI for social good Project (2020) https://GitHub.com/ASL** • **Listed by Software Freedom Foundation (now Digital Freedom Foundation) as 2006 Best global Software Freedom Day (SFD) team (2006)** • **Third position in Technical project presentation for Abadina College's first time at Junior Engineer and Technical Students (JETS) competition at state level. (1994)**
Professional Competence Certifications	Udacity nanodegree on Intel Edge AI for IOT Developer & AWS ML (in view), IOT and ARDUINO. Certified Lean Six Sigma Black belt (CSSBB),Project Management-PMP in view, Auditor (QMS, EHS),COREN, NSE, ASME-USA, ImechE(in view)-UK.
References	❖ **Prof. O. Bamiro**, former Vice **Chancellor**, University of Ibadan, Ibadan, Nigeria. **Tel: +2348023151513;** oabamiro@yahoo.com ❖ **Prof Fadi AL-Turjman,** Asst Dean, Director of Research center for AI and Robotics Institue,Near East University, **Tel**: **+905428520985** fadi.alturjman@neu.edu.tr ❖ **Asst. Prof. Dr. Auwalu Saleh Mubarak,** Lecturer, Research center for AI and Robotics Institue,, Near East University, **Tel**: **+905338717889** auwalusaleh.mubarak@neu.edu.tr ❖ **Dr. Seun Kolade**, Faculty Head of Doctoral Training Programme, De Montfort University, Leicester, UK. **Tel**: **+447897265890** seunkolade2014@gmail.com ❖ **Mr. Moses Duphey, World Bank** ,P.O Box C847, Accra, Ghana, **Tel:+233244649748**; **mosesduphey@yahoo.co.uk**