applied sciences

MDPI

Article

Short-Term Energy Consumption Forecasting Analysis Using Different Optimization and Activation Functions with Deep Learning Models

Mehmet Tahir Ucar [1,*] and Asim Kaygusuz [2]

[1] Ergani Vocational School, Dicle University, 21280 Diyarbakır, Türkiye
[2] Department of Electrical and Electronics Engineering, Engineering Faculty, Inonu University, 44280 Malatya, Türkiye; asim.kaygusuz@inonu.edu.tr
* Correspondence: mtahir.ucar@dicle.edu.tr

Abstract: Modelling events that change over time is one of the most difficult problems in data analysis. Forecasting of time-varying electric power values is also an important problem in data analysis. Regression methods, machine learning, and deep learning methods are used to learn different patterns from data and develop a consumption prediction model. The aim of this study is to determine the most successful models for short-term power consumption prediction with deep learning and to achieve the highest prediction accuracy. In this study, firstly, the data was evaluated and organized with exploratory data analysis (EDA) on a ready dataset and the features of the data were extracted. Studies were carried out on long short-term memory (LSTM), gated recurrent unit (GRU), simple recurrent neural networks (SimpleRNN) and bidirectional long short-term memory (BiLSTM) architectures. First, four architectures were used with 11 different optimization methods. In this study, it was seen that a high success rate of 0.9972 was achieved according to the R^2 score index. In the following, the first study was tried with different epoch numbers. Afterwards, this study was carried out with 264 separate models produced using four architectures, 11 optimization methods, and six activation functions in order. The results of all these studies were obtained according to the root mean square error (RMSE), mean absolute error (MAE), and R^2_score indexes. The R^2_score indexes graphs are presented. Finally, the 10 most successful applications are listed.

Keywords: deep learning models; electricity consumption prediction; exploratory data analysis (EDA); long short-term memory (LSTM); gated recurrent unit (GRU); simple recurrent neural networks (Simple RNN); bidirectional long short-term memory (Bi-LSTM); optimization and activation functions

check for updates

Academic Editor: Seung-Hoon Yoo

Received: 2 May 2025
Revised: 8 June 2025
Accepted: 11 June 2025
Published: 18 June 2025

Citation: Ucar, M.T.; Kaygusuz, A. Short-Term Energy Consumption Forecasting Analysis Using Different Optimization and Activation Functions with Deep Learning Models. *Appl. Sci.* **2025**, *15*, 6839. https://doi.org/10.3390/app15126839

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Increasingly, information communication technologies and artificial intelligence applications are being used in smart grids. Networks are being built healthier, faults are better detected and resolved, needs are better identified, outages are reduced, forecasts are improved, illegal use is being reduced, and quality is being improved. In the infrastructure of forward-looking programs, new inferences and calculations are being made in light of previous information and planning is improving as a result. The inferences, predictions, and calculations made for future situations, which we call predictions, actually form the basis of all the planning we do.

Forecasting is an indispensable element of planning. Successful and consistent planning depends on the success of forecasts, predictions, and calculations. The accuracy of the prediction is a very important issue and affects the success of the programming. Therefore, it is important for programmers to focus on prediction, to specialize in it, and to generate and develop new methods in this area.

Studies in the field of forecasting are very diverse. The majority of these studies have focused on stock market forecasting [1–4]. In addition, Cheng et al. worked on a stock-trading system [5], Vargas et al. on predicting the intraday movement direction of a selected index [6], and Li et al. on air temperature forecasting [7]. In addition, studies on the prediction of meteorological conditions such as air temperature, humidity, solar radiation values, air pollution, etc., are also noteworthy [8–11].

In the field of electricity, there are studies on remaining useful life (RUL) prediction and degradation process prediction [12–15]. When we look at the field of smart grids, there are studies such as energy-consumed estimation, energy-to-be-produced estimation, and solar or wind potential estimation [16–18].

In electricity planning, the planning of electrical energy for the following hours and days is very important. Incorrect planning, interruptions, bottlenecks, power fluctuations, and similar negativities directly cause great damage to production, the economy, and the country.

If power plants can estimate the amount of energy they will produce or sell, it will be easier for load dispatch units to make programming for the following days. Thus, the demanded and produced energy estimates will be balanced and will meet each other, and unplanned energy outages, voltage fluctuations, and system crashes will be prevented. In addition, these estimates are important in the operation, maintenance, fuel use, and similar planning of electricity production facilities and in electricity energy pricing. It should not be forgotten that increasing electricity efficiency will have important effects on the development of the country. Therefore, making the plans and estimates with high accuracy and minimum error rates will direct the measures to be taken and the studies to be carried out.

We can list electricity forecasts under four headings. 1. Very short-term load forecast (VSTLF), 2. Short-term load forecast (STLF), 3. Medium-term load forecast (MTLF), and 4. Long-term load forecast (LTLF). This study is based on short-term electricity consumption forecasts.

VSTLF requires only historical loads; STLF usually requires historical loads and weather information; MTLF requires weather and economic information; and LTLF needs weather, economic, demographic, and sometimes land-use information [19].

2. Literature Review

While statistical methods work well under normal conditions, their performance decreases during changing weather conditions, different sociological and economic conditions, and holidays. For this reason, researchers have turned to new approaches, and artificial intelligence methods have gained importance. The main artificial intelligence methods used for load forecasting are artificial neural networks (ANN), fuzzy logic, support vector machines (SVM), support vector regression (SVR), genetic algorithms (GA), particle swarm optimization (PSO), and ant colony optimization (ACO) [20–22].

Xiaoou Monica Zhang et al. and Yang Meiyan et al. used an SVM modeling approach as an energy consumption predicting algorithm. Here, analyses were made according to weather conditions, calendar, and usage time prices. The analysis shows that estimating residential energy consumption based on weather, calendar, and time-of-use price is feasible

and reliable with sufficient accuracy for daily or hourly estimation for some individual residential uses [23,24].

According to the literature, artificial intelligence methods are more successful than statistical methods in capturing different features that affect the prediction and in making more accurate predictions.

In time-series studies, artificial intelligence methods show success despite various limitations such as the poor fit of statistical forecasting models to nonlinear data, the sensitivity of model parameters, and low model generalization ability. However, most methods typically adopt a predefined nonlinear shape and cannot simulate the real nonlinear relationship [25].

Although machine learning methods were known in the field of artificial intelligence, they were not widely used and were not preferred due to a lack of data. However, with the use of smart meters and their widespread use, a lot of data flow has occurred. As a result, machine learning methods began to be used instead of previous methods and they began to yield successful results.

In fact, thanks to the recorded data and developing data science, deep learning, a sub-branch of machine learning, has developed rapidly and has surpassed traditional machine learning in terms of prediction accuracy and efficiency in some areas [26].

An extensive literature study has also been conducted on artificial intelligence, machine learning, and deep learning applications [27].

There are different architectures designed to solve different problems in deep learning. The most well-known of these architectures are auto-encoders, convolutional neural networks (CNN), recurrent neural networks (RNN), long short-term memory (LSTM), restricted Boltzmann machines (RBM), deep belief networks (DBN), generative adversarial networks (GAN), transfer learning, and deep reinforcement learning (DRL).

In deep learning, RNNs, which process sequences with temporal relationships using self-feedback neurons, are used specifically for time series data [28]. Although the power consumption model is determined by long-term dependence, the predicted value of RNN is only related to the last value. The vanishing of gradients or exploding occurs frequently, which reduces the prediction accuracy. In recent years, gated cyclic unit LSTM, a sub-branch of RNN, has been frequently used to solve this problem [25].

LSTM has established itself in most time series forecasting studies and has outperformed other short-term load forecasting algorithms used in related studies [29–31].

Some studies have shown that deep learning-based algorithms such as the LSTM outperform traditional-based algorithms such as the autoregressive integrated moving average (ARIMA) model [32,33].

As seen in the literature, the LSTM model provides suitable solutions for time series. For this reason, LSTM has been mostly used in deep learning studies on time series and electrical energy forecasting studies in recent years. However, researchers were not satisfied with this performance and tried to increase the performance by playing with the LSTM model, forecast data, training method, hyper parameters and other variables. In our work, we use a total of four different architectures, including SimpleRNN and LSTM, as well as GRU and BiLSTM. We were not satisfied with the architectures we chose, but attempted to see the effects of these variables on the system and increase the system performance by playing with hyper parameters and other variables. Briefly, we examined the studies conducted to increase performance over the LSTM architecture and use different architectures:

Lin et al. used a two-stage attention-based LSTM network for short-term regional load probability forecasting, taking into account feature correlation and temporal dependencies. In the first stage, a feature attention-based encoder was built to calculate the correlation of

input features with the electric charge at each time step. The most relevant input features were adaptively selected. In the second phase, a temporal attention-based decoder was developed to investigate time dependencies. Then, an LSTM model integrated these attention results, and probabilistic predictions could be made using a pinball loss function [34].

Ding et al. proposed an evolutionary dual attention-based long short-term memory model and introduced binary features using feature combination. This study compared the prediction performance of eight prediction methods. The results show that an attention mechanism can improve the efficiency of the LSTM algorithm when the model uses input time series data [25].

Wang et al. proposed a new approach based on an LSTM network to predict periodic energy consumption. First, hidden features were extracted by an autocorrelation plot among real industrial data. Correlation analysis and mechanism analysis contributed to finding appropriate secondary variables as model inputs. In addition, the time variable was complemented to fully capture periodicity. Experimental results on a specific cooling system show that the proposed method has higher prediction performance compared with back propagation neural network (BPNN), autoregressive and moving average (ARMA), and autoregressive fractionally integrated moving average (ARFIMA) [35].

A different model is gate recurrent unit networks (GRU). In some studies, GRU is considered to provide more accuracy, improve prediction, produce better performance results, and be faster than LSTM [36–39].

Another method to be evaluated is bidirectional LSTM (Bi-LSTM). Bi-LSTMs provide additional training by moving the input data twice (i.e., 1 left to right and 2 right to left). Siami-Namini et al. show that Bi-LSTM-based models provide better predictions than normal LSTM-based models. More specifically, Bi-LSTM models provide better forecasts than ARIMA and LSTM models. It has also been observed that Bi-LSTM models reach equilibrium much slower than LSTM-based models [40]. In another short-term energy consumption study, it was seen that the BiLSTM method is more successful than the SVR, CNN, and GRU methods [41].

There are also some studies on short-term electricity load consumption forecasting that use four of the LSTM, GRU, Simple RNN, and Bi-LSTM models. However, it can be seen that similar studies are mostly conducted in the fields of pandemics, production forecasting, and stock markets. In [42], a study was conducted on deaths and recoveries in ten major countries affected by COVID-19, and the performance of the models was measured using root mean squared error (RMSE), mean absolute error (MAE), and R^2_score indices. In most cases, the Bi-LSTM model outperforms in terms of validated indices. The models ranked from best to worst performance in all scenarios were Bi-LSTM, LSTM, GRU, SVR, and ARIMA. In [43], a price forecasting system was developed by comparative analysis between LSTM, GRU, and Bi-LSTM models. It can be seen that the Bi-LSTM model, which used the last 5 days of trading data of the stock as input, reached an accuracy value of 63.54%. It can be seen that the best result is obtained with this method.

If we perform a short meta-analysis and look critically at three similar studies using the same dataset, we can see the following:

In a study using the same dataset as the one we studied, it was seen that missing and excess data in the dataset were not analyzed, and the data was not sorted according to time. Despite this, it can be seen that a successful study was performed with a multi-layered and complex Simple RNN and LSTM architecture with an R^2 metric of 0.9814 [44]. Although we used a simple model with a single hidden layer in our study, 114 different models exceeded this value.

In another study using the same dataset, missing and excess data in the dataset were not analyzed. The models were compared using mean squared error (MSE) as the loss

function. Here, the lowest error rate was shown as RNN 0.4, LSTM 0.31, and GRU 0.24. In our study, it was observed that this metric drops to a value of 0.002 [45].

A third study using the same dataset analyzed missing and excess data in the dataset. Models were compared using MSE, MAE, and RMSE as loss functions. Here, in the study conducted with the MSE loss function, the lowest error rate belongs to LSTM with 0.103. The order of success for the other models is multi-layer perceptron (MLP), CNN, linear regression, SVR, decision tree, extreme learning machine (ELM), and GRU [46].

It has been observed that studies using LSTM, GRU, and Bi-LSTM architectures, which are a sub-branch of RNN that has recently been studied extensively in the literature, have yielded successful results. Therefore, in this paper, Simple RNN, LSTM, GRU, and Bi-LSTM models were used and an increase of the prediction performance was attempted with different hyper-parameter values. Experiments were also conducted with many hyper-parameters that are not commonly seen in the literature.

The next part of the paper is organized as follows: Section 3 presents the models, methods, and aspects that differentiate the proposed study from other studies. Section 4 presents the findings obtained according to the methods used. Section 5 presents the findings and results of our study and visualizes them with graphs. Section 6 ends with the conclusion.

3. Materials and Methods

In this study, estimation studies are carried out using deep learning algorithms, which is a sub-branch of machine learning, from different artificial intelligence methods whose performance has been tested in the literature in estimating electricity consumption. It is seen that in most of the studies, joint studies such as electrical energy demand production or consumption estimation are carried out. However, generalizable methods have not yet been achieved in this regard, and even though some methods have been applied to similar datasets, different results have been obtained. In other words, new and different applications and experiments are often needed. There are also many areas that have not yet been clearly studied and are waiting to be developed. Therefore, there is a need to compare different techniques and produce different models in order to make model recommendations for datasets.

Some studies were limited because they were carried out without organizing the dataset and because they used classical methods for hyper-parameters, which have been found to be successful before. Developing models by considering these constraints will constitute the most important and different part of the study.

The studies that make the study different, meaningful and subjective are listed below.

1. Data will be organized.
2. 4 different architectures will be used.
3. To The effect of epoch numbers on the study will be examined.
4. 6 different methods will be used as activation functions.
5. 11 different methods will be used as optimization functions.

Using the above 5 methods, different models will be created in the field of deep learning and hourly energy consumption estimates will be made.

As data, hourly energy consumption data in megawatts for 17 years (1 January 2002–3 August 2018) of PJM, a regional transmission organization in the USA, will be used.

First, the data will be examined and organized. Excess data will be removed and missing data will be completed.

In addition to consumption values, new columns are created for year, month, week, day, and hour values. Again, 24-h data for each day in the day-ahead market is taken and used in the next hour's consumption forecast. In this way, it is aimed to implement

a supervised learning step and increase success. Again, with this method, it is aimed to increase our single-column consumption data to 6 columns and to reach 144 columns when 24-h sample values are taken. Thus, our dataset is intended to consist of 145 columns and 145,368 rows, including the datatime column. With this technique, our simple, single-column data is adapted to the MLP model.

In normal studies, the data will be in two parts. The first part will be subjected to the training process (80%), and the second part (20%) will be used for the test performance of these trained models. In this way, models will be trained with the training set, and their performance will be observed by making predictions on the test data.

As a method, RNN, a model designed to work with time series in the field of deep learning, LSTM, GRU, and Bi-LSTM architectures, which are a sub-branch of RNN, are used, and the results are compared. It is seen that in most studies, restrictions are imposed on the dataset and hyper-parameters. Developing models by considering these constraints and comparing the results by changing these hyper-parameters are among the main topics of this study. For this purpose, experiments are carried out with different activation functions for each model. For each model, 10, 50, and 100 epoch trials are performed, and the results are interpreted. In addition, a comparative analysis is made between the models by using different activation functions and optimization methods for each model. Finally, the 10 models that provided the highest performance in all tests conducted with 50 epochs are presented.

3.1. Data Analysis

3.1.1. Data

For the purpose of this article, we used hourly energy consumption data for the years 2002–2018, which is freely available on Kaggle and belongs to PJM Interconnection LLC (PJM) (Norristown, PA, USA). The datasets are hourly energy consumption data from PJM, a regional transmission organization in the United States that supplies electricity to Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, New Jersey, North Carolina, Ohio, Pennsylvania, Tennessee, Virginia, West Virginia, and the District of Columbia. We did not use the entire dataset but a part of it in our study. The data we use is the energy consumption rate of PJM East Region (PJME). PJM Interconnection LLC (PJME) provides a time series dataset consisting of 145,366 rows and 2 features. The consumption data is in megawatts (MW) [47].

3.1.2. Exploratory Data Analysis

Exploratory data analysis (EDA) includes the presentation, summarization, and graphical presentation of these analyses in an understandable and easy way in order to obtain relevant information from existing data by utilizing probability and statistical methods. These studies provide the foundations of feature engineering. That is, it is the work of creating, transforming, and inferring features from the dataset so that the model can work in the most efficient way possible.

First, a clear exploratory data analysis template is created that extracts and summarizes the most important features of the dataset and focuses on time series. Some common Python libraries 3.8.20, such as Pandas 1.4.4, Matplotlib 3.6.2, Numpy 1.19.2, Seaborn 0.12.2 and Statsmodels 0.13.5, are used for this.

The first thing to do when working with time series is to roughly observe and check the type of data, its values, and whether there are any incorrect or missing values. One of the important things we do is plot the data. Thus, many features such as graphs, patterns, unusual observations, changes over time, and relationships between variables can be observed here. The results from these observations should be incorporated into the forecast-

ing model as much as possible. In addition, some mathematical tools, such as descriptive statistics and time series decomposition, also provide us with great benefits [48,49].

In the next section, studies and conclusions regarding some of the steps implemented for EDA will be made.

Descriptive Statistics, Data Organization, and Time Plot

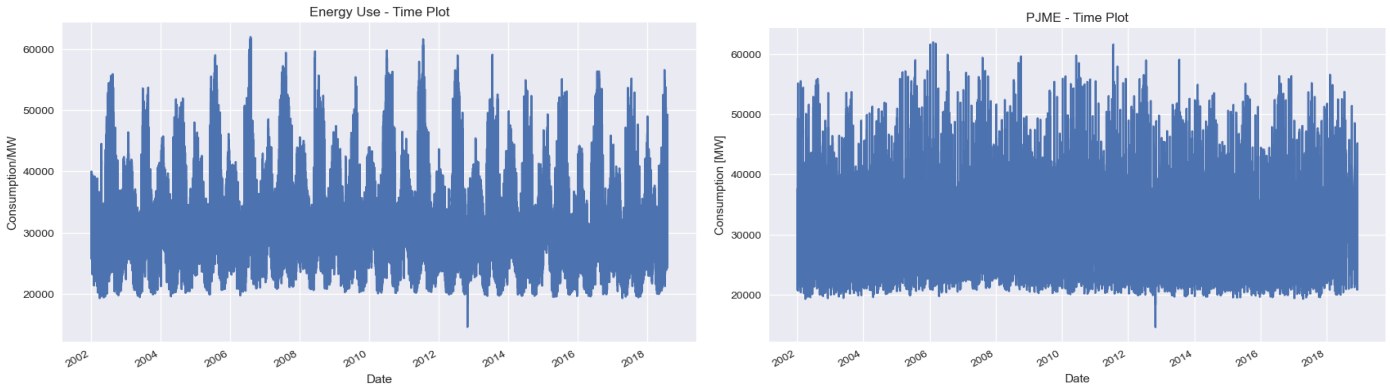
The first and last 5 rows of our raw data are shown in Table 1a. Our data consists of two columns: datetime and consumption amount.

Table 1. (**a**) First and last 5 rows of unedited raw data, (**b**) First and last 5 rows of edited data.

(a)			(b)		
Line No	**Date Time**	**PJME_MW**	**Line No**	**Date Time**	**PJME_MW**
1	2002.12.31 01:00	26,498.0	1	1.01.2002 01:00	30,393.0
2	2002.12.31 02:00	25,147.0	2	1.01.2002 02:00	29,265.0
3	2002.12.31 03:00	24,574.0	3	1.01.2002 03:00	28,357.0
4	2002.12.31 04:00	24,393.0	4	1.01.2002 04:00	27,899.0
5	2002.12.31 05:00	24,860.0	5	1.01.2002 05:00	28,057.0
145362	2018.01.1 20:00	44,284.0	145388	2.08.2018 20:00	44,057.0
145363	2018.01.1 21:00	43,751.0	145389	2.08.2018 21:00	43,256.0
145364	2018.01.1 22:00	42,402.0	145390	2.08.2018 22:00	41,552.0
145365	2018.01.1 23:00	40,164.0	145391	2.08.2018 23:00	38,500.0
145366	2018.01.2 00:00	38,608.0	145392	3.08.2018 00:00	35,486.0

As can be seen from Table 1a, the values are sorted by years and hours but not by months and days. The data is not ordered from past to future. Moreover, our data consists of 145,366 rows and 2 columns excluding the header row. As a result of the examinations made according to time, it can be observed that 30 lines are missing and 4 lines are repeated. Missing values were added by calculating the mean value. The more appropriate of the redundant lines was selected and the other line was deleted. The data was then sorted from the past to the future. The final version of our data is shown in Table 1b.

Figure 1a shows the time-dependent change in consumption according to raw data, and Figure 1b shows the time-dependent change in consumption according to edited data. Here it can be seen that unedited raw data will mislead us. Because incomplete and unordered data will shift both the peak time and the holiday hours.



(**a**) (**b**)

Figure 1. (**a**) Unedited raw data time plot, (**b**) Edited raw data time plot.

In Figure 1b, there is no major increase/decrease trend over the years and the average consumption remains more stable compared with Figure 1a.

Seasonal Plots

When we plot graphs annually, monthly, weekly, and daily, the recurring events or differences that we call seasonality become clearer.

According to Figure 1b, it can be seen that all years have similar patterns. It can be observed that consumption values increase significantly in winter and summer (for heating/cooling purposes) and reach the highest levels in summer. However, in the mild spring and autumn months, consumption can be seen to be at a minimum.

As seen in Figures 2 and 3, consumption values reached maximum values in the 7th and 8th months and minimum values in the 4th and 10th months.

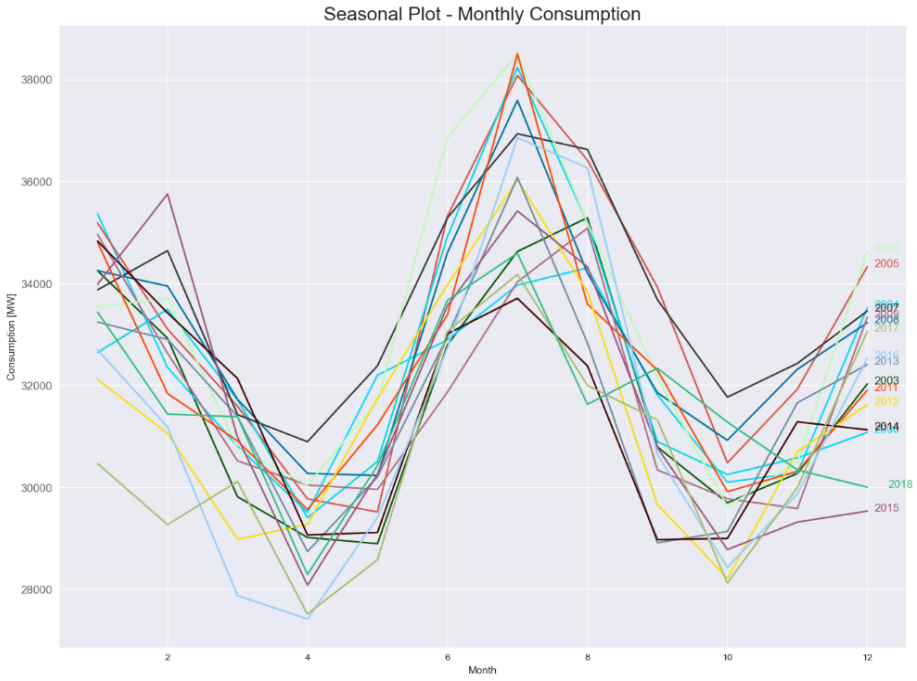


Figure 2. PJME yearly–monthly consumption seasonal plot.

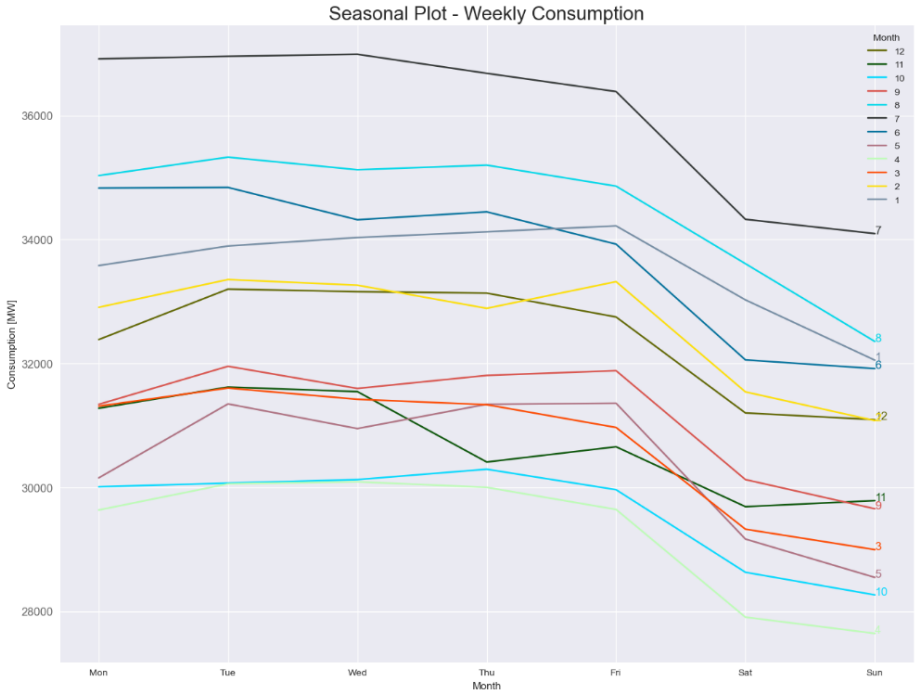


Figure 3. PJME monthly–weekly–daily consumption seasonal plot.

Figure 4 shows that consumption on weekdays is higher than consumption on weekends. Moreover, the highest difference occurs during daytime hours. The highest consumption is seen at 18:00 and 19:00 (peak hours), while the lowest consumption is seen at 3:00, 4:00, and 5:00 (midnight).

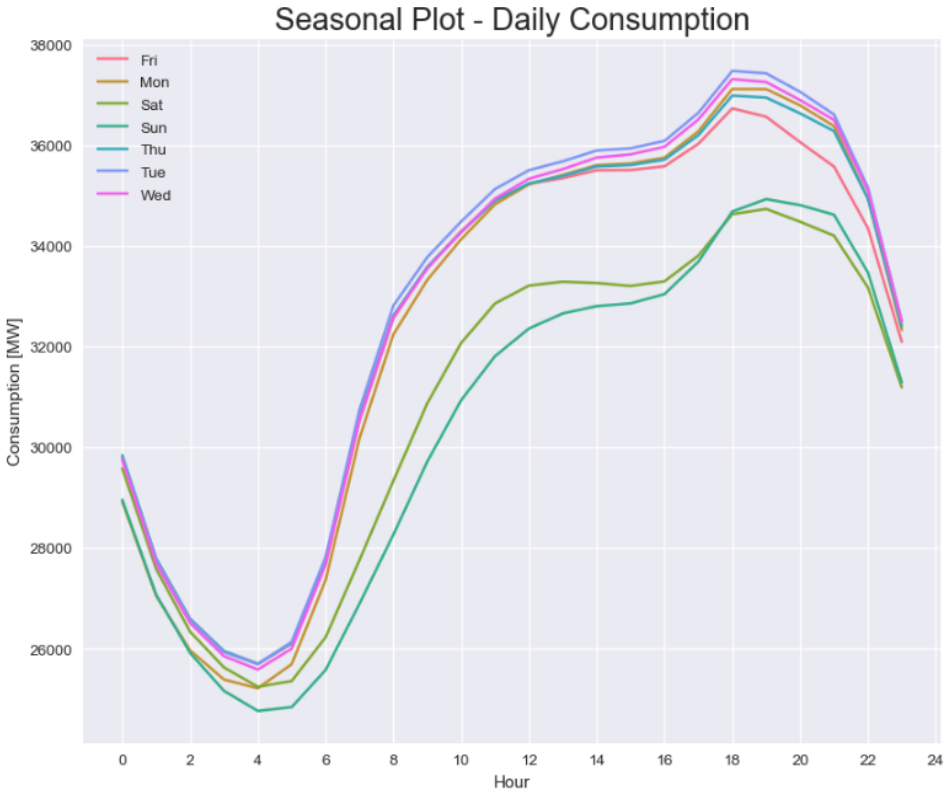


Figure 4. PJME daily–hourly consumption seasonal plot.

The values to be considered for time series are the time intervals that exhibit seasonal behavior and the most variability. For this reason, the values that should be taken as an example in our study are seen as the last 24 h values, because, when looking at the annual, monthly, weekly, daily, and hourly values in Figures 2–4, it can be seen that the data shows similar behavior and there is no extreme difference between the values. The maximum difference is seen between intraday hourly values. However, for example, the consumption value of the previous day at 11:00 and the consumption value of the next day at 11:00 are similar. Therefore, when estimating today's consumption, one should look at yesterday as the most similar value.

According to Figures 2–4, it can be seen that year, month, week, day, and hour affect consumption values. Therefore, when estimating consumption, this information should be converted into columns and used when estimating. That is, the columns to be analyzed should be year, month, week, hour, day, and PJME_MW columns. Accordingly, the first 5 rows of the final version of our data are shown in Table 2.

Table 2. First 5 rows of edited data.

Date Time	PJME_MW	Year	Month	Week	Hour	Day	day_str	Year_Month
1.01.2002 01:00	30,393.0	2002	1	1	01:00:00	1	Tue	2002_1
1.01.2002 02:00	29,265.0	2002	1	1	02:00:00	1	Tue	2002_1
1.01.2002 03:00	28,357.0	2002	1	1	03:00:00	1	Tue	2002_1
1.01.2002 04:00	27,899.0	2002	1	1	04:00:00	1	Tue	2002_1
1.01.2002 05:00	28,057.0	2002	1	1	05:00:00	1	Tue	2002_1

3.2. Deep Learning Study

Some of the parameters we will use in our deep learning model are shown below.

SEED = 123
batch_size = 128
return_sequence = True
target = 'PJME_MW' *
n_hours = 24 **
train_size = 0.8 ***
cols_to_analyze = ["PJME_MW", "year", "month", "week", "day", "hour"]

(* We bought one dimension as a basis, we could have taken more than one dimension. ** The value we will take for sampling. *** 80% of the data will be used for training and 20% for testing.)

In order to produce the same values every time, we set the fixing value, seed, equal to the seed value we determined before, i.e., 123.

We are trying to estimate the energy consumption for the next hour. Of course, the energy consumption values of the previous day will be important for us. Past values are known as lag. The value at time t is greatly affected by the value at time t-1. This application is called framing in the field of feature engineering. In our study, 24 h data for each day in the day-ahead market is taken and used to forecast consumption for the next hour. In other words, the aim is to predict the 25th hour with 24 h of historical data (with n_hours = 24). Then, we need a frame that estimates the 26th hour from the past 2–25 h. Our time series columns are rearranged based on the year, month, week, day, hour, and consumption value in Table 2. For the 6 columns in our time series, all values from the previous day t-1 to t-24 are taken and included in the calculation. When we take the 24 h values of the 6 columns and arrange them side by side, 144 columns are formed. Thus, our dataset is intended to consist of 145 columns and 145,368 rows, including the datatime column. With this technique, our simple single-column data was adapted to the MLP model. For this, a supervised learning method is applied by using a ready-made function.

In addition to the libraries given in the previous sections for deep learning, some common Python libraries, such as Scikit-learn 1.6.1, TensorFlow 2.3.0, and Keras 3.10.0, are also needed.

The following model is used as a model.

```
model = Sequential ()
model.add (LSTM (50, activation = 'relu',input_shape = (X_train.shape [1], X_train.shape [2])))
model.add (Dense (1))
model.compile (optimizer = 'adam', loss = 'mse', metrics = ['accuracy'])
```

The next part is continued through the program, and as a result of changing some parameters, the performance of the models can be measured with MAE, RMSE, and R^2_score indices.

3.3. Creating the Program Interface

First of all, the Python program was used in all studies via the Jupyter Integrated Development Environment (IDE) within Anaconda. The graphs in the 3rd section were drawn with the Python program, and the graphs in the 5th section were drawn with the Microsoft Excel program. Additionally, Microsoft Excel was used in some studies and tables. RMSE, MAE, and R^2 metrics were calculated via Python.

4. Working

Within the scope of study, 32 different group studies consisting of 352 trials were conducted. The studies were carried out with 44 times 10 epochs, 11 times 70 epochs,

33 times 100 epochs, and 264 times 50 epochs. In addition, the computer was turned off and on again during each study to ensure that what was learned in the previous study would not affect the next study and that the RAM energy was completely discharged.

LSTM, GRU, Simple RNN, and Bi-LSTM models were used in all studies, respectively. Along with four different models, 11 different optimization methods belonging to the Keras library were tried for optimization. These were Adam, Nadam, Adamax, RMSprop, RMSprop*, SGD, SGDtrue, SGDfalse, Adagrad, FTRL, and Adadelta methods. A few more optimization models were tried but were not added to the lists because they did not yield results. The seven most successful of these methods are painted in their own unique colors. Default values for standard methods were used in this study. However, some values for the RMSprop*, SGDtrue, and SGDfalse methods have been changed. The details of these three methods are listed in Table 3.

Table 3. Details of the methods used.

Method	How to Use in the Program
RMSprop*	optimizer = tf.keras.optimizers.RMSprop (learning_rate = 0.01, rho = 0.9)
SGDTrue	optimizer = tf.keras.optimizers.SGD (lr = 0.01, decay = 1×10^{-5}, momentum = 0.9, nesterov = True)
SGDFalse	optimizer = tf.keras.optimizers.SGD (lr = 0.001, decay = 1×10^{-5}, momentum = 1.0, nesterov = False)

4.1. Analysis of Consumption Forecast Using 11 Different Optimizer Methods on Four Different Models

Using four different deep learning models, LSTM, GRU, SimpleRNN, and BiLSTM, 11 different optimization methods were tested for each model. The results of this study were measured with RMSE, MAE, and R^2 metrics and are given in Table 4. In the table, the models are listed from top to bottom according to their success status. In these experiments, the epoch value was taken as 50.

Table 4. Four model–11 optimizer study results. The background of values with an R^2 value of 0.99 and above is painted green, and the background of values between 0.98 and 0.99 is painted pink. The background of values with an R^2 value of 0 or negative is painted brown.

Model	Optimizer	test_rmse	test-mae	test_R^2	Model	Optimizer	test_rmse	test-mae	test_R^2
LSTM(50, activation = 'relu'	Adam	339.33	242.54	0.9972	SimpleRNN(50, activation = 'relu'	Nadam	368.15	265.96	0.9968
	Nadam	428.91	321.56	0.9954		Adam	425.05	307.47	0.9956
	RMSprop	420.57	288.89	0.9957		RMSprop	486.2	357.39	0.9942
	RMSprop*	500.28	376.88	0.9938		adamax	567.34	440.71	0.9921
	adamax	517.6	406.57	0.9935		SGDTrue	667.51	503.7	0.9889
	SGDTrue	768.55	586.43	0.9853		RMSprop*	812.6	664.71	0.9832
	SGD	1602.39	1284.1	0.9294		SGD	904.34	684.67	0.9792
	adagrad	2278.76	1838.13	0.8518		adagrad	1857.1	1488.04	0.9031
	Ftrl	5526.37	4603.25	−5.2028		Ftrl	3173.83	2492.14	0.4742
	adadelta	6749.2	5649.35	−5.44		adadelta	5071.94	4243.3	−0.5113
	SGDFalse	13,125.06	11,938.01	-1×10^{13}		SGDFalse	17,577.15	16,466.97	-2×10^{12}
GRU(50, activation = 'relu'	Adam	412.18	296.07	0.996	Bidirectional(LSTM(50, activation = 'relu'	Adam	313.04	223.66	0.9976
	Nadam	404.5	296.4	0.996		Nadam	345.03	255.65	0.9971
	adamax	481.16	366.41	0.9945		adamax	458.3	342.28	0.9948
	RMSprop	474.57	347.39	0.9944		RMSprop	564.17	458.31	0.9923
	RMSprop*	659.23	537.33	0.9887		RMSprop*	655.86	518.52	0.989
	SGDTrue	961.56	721.47	0.9754		SGDTrue	831.81	635.59	0.9829
	SGD	1336.02	1082.17	0.9546		SGD	1480.73	1165.06	0.9419
	adagrad	1935.03	1535.2	0.8824		adagrad	2212.38	1727.57	0.865
	Ftrl	4800.64	3921.81	−1.7584		Ftrl	4940.55	4151.66	−2.66
	adadelta	6497.18	5390.8	−5.6841		adadelta	6296.76	5378.12	−8.6355
	SGDFalse	12,246.21	10,396.46	-4×10^{13}		SGDFalse	11,978.96	10,797.69	0

4.2. 4 Analysis of Consumption Forecast Using 10 Epochs, 50 Epochs, and 100 Epochs Using 11 Different Optimizer Methods on Four Different Models

Using four different deep learning models, LSTM, GRU, SimpleRNN, and BiLSTM, 11 different optimization methods were tested with 10 epochs, 50 epochs, and 100 epochs. However, in the study conducted with the BiLSTM model, the computer gave an error before reaching the 100-epoch value. Therefore, experiments were conducted with 70 epochs instead of 100 epochs for the BiLSTM model. This study was measured with RMSE, MAE, and R^2 metrics, and the results are given in Table 5. The values in the table are listed from top to bottom according to their success status for each model. In all experiments in this section, ReLU was used as the activation function. Table 5 includes Table 4 for comparisons.

Table 5. Four models–11 optimizers 10–50–100 epoch test results. The background of values with an R^2 value of 0.99 and above is painted green, and the background of values between 0.98 and 0.99 is painted pink. The background of values with an R^2 value of 0 or negative is painted brown.

Epochs		10 Epochs			50 Epochs			100 Epochs		
Model	**Optimizer**	**test rmse**	**test mae**	**test_R^2**	**test rmse**	**test mae**	**test_R^2**	**test rmse**	**test mae**	**test_R^2**
LSTM(50, activation = 'relu'	Adam	571	448	0.9918	339	243	0.9972	289	208	0.9979
	Nadam	660	515	0.9885	429	322	0.9954	311	229	0.9976
	RMSprop	810	653	0.9838	421	289	0.9957	536	430	0.9931
	RMSprop*	828	691	0.9819	500	377	0.9938	2602	2198	0.7719
	adamax	1282	985	0.9542	518	407	0.9935	345	256	0.9971
	SGDTrue	1727	1348	0.9095	769	586	0.9853	607	448	0.9909
	SGD	2219	1792	0.8494	1602	1284	0.9294	965	753	0.9761
	adagrad	4293	3464	−0.6808	2279	1838	0.8518	2076	1678	0.8796
	Ftrl	6401	5272	−8.7777	5526	4603	−5.2028	4754	3896	−1.4893
	adadelta	9462	7775	−2.6307	6749	5649	−5.44	4214	3415	−0.5044
	SGDFalse	6851	6365	−0.2154	13,125	11,938	-1×10^{13}	33,342	32,704	−7.4604
GRU(50, activation = 'relu'	Adam	601	472	0.991	412	296	0.996	339	237	0.9972
	Nadam	643	488	0.9896	405	296	0.996	358	266	0.9969
	adamax	908	707	0.9787	481	366	0.9945	418	308	0.9959
	RMSprop	729	587	0.9868	475	347	0.9944	538	431	0.993
	RMSprop*	826	625	0.9809	659	537	0.9887	690	564	0.9873
	SGDTrue	1328	1033	0.9534	962	721	0.9754	733	546	0.9862
	SGD	1725	1378	0.9207	1336	1082	0.9546	1173	942	0.965
	adagrad	3868	3063	−0.3151	1935	1535	0.8824	1713	1386	0.9248
	Ftrl	6174	5109	−9.9445	4801	3922	−1.7584	3567	2837	0.1859
	adadelta	7176	5724	−4.2308	6497	5391	−5.6841	4605	3768	−1.6678
	SGDFalse	12,883	11,131	-4×10^{13}	12,246	10,396	-4×10^{13}	47,910	47,468	-3×10^{14}
SimpleRNN(50, activation = 'relu'	Nadam	583	444	0.9912	368	266	0.9968	397	300	0.9962
	Adam	765	592	0.985	425	307	0.9956	396	275	0.9962
	RMSprop	926	786	0.9783	486	357	0.9942	440	306	0.9953
	adamax	789	602	0.9844	567	441	0.9921	416	310	0.9958
	SGDTrue	930	716	0.9779	668	504	0.9889	615	457	0.9906
	RMSprop*	832	615	0.9814	813	665	0.9832	1150	941	0.963
	SGD	1640	1300	0.9215	904	685	0.9792	744	555	0.9861
	adagrad	2566	2035	0.7921	1857	1488	0.9031	1550	1242	0.9355
	Ftrl	5403	4469	−3.4172	3174	2492	0.4742	1673	1350	0.9184
	adadelta	9941	8009	−0.7133	5072	4243	−0.5113	2686	2132	0.7625
	SGDFalse	14,965	13,796	-4×10^{15}	17,577	16,467	-2×10^{12}	19,404	18,287	0
Epochs		**10 Epochs**			**50 Epochs**			**70 Epochs**		
Bidirectional(LSTM(50, activation = 'relu'	Adam	532	404	0.9929	313	224	0.9976	321	228	0.9975
	Nadam	563	438	0.992	345	256	0.9971	337	253	0.9972
	adamax	749	614	0.9855	458	342	0.9948	432	320	0.9954
	RMSprop	748	597	0.9857	564	458	0.9923	486	344	0.9943
	RMSprop*	893	754	0.9796	656	519	0.989	error	error	error
	SGDTrue	1381	1071	0.9479	832	636	0.9829	718	544	0.9873
	SGD	2036	1568	0.8873	1481	1165	0.9419	1357	1072	0.9515
	adagrad	3599	2861	0.0957	2212	1728	0.8649	2117	1649	0.8806
	Ftrl	6267	5193	−9.7917	4941	4152	−2.66	4371	3639	−0.8450
	adadelta	10,693	8918	−3.7992	6297	5378	−8.6355	4993	4202	−2.7761
	SGDFalse	10,526	9370	-7×10^{15}	11,979	10,798	0	18,981	17,837	-4×10^{14}

4.3. Analysis of Consumption Forecast Using Four Different Models, Six Different Activation Functions, and 11 Different Optimization Methods

In our study, four different models (LSTM, GRU, Simple RNN, and BiLSTM), six different activation functions (ReLU, tanh, ELU, LeakyReLU, sigmoid, softmax), and 11 different optimization methods were tested. In each study, one of the four different models, six different activation functions, and 11 different optimization methods were changed and 264 different models were studied. The epoch value was taken as 50 in the experiments. The results of this study were measured with RMSE, MAE, and R^2 metrics and the results are given in Table 6. The values in the table are listed from top to bottom according to their success status for each model. Table 6 includes Table 4 for comparisons.

Table 6. Four models–six activation functions–11 optimizers study results. The background of values with an R^2 value of 0.99 and above is painted green, and the background of values between 0.98 and 0.99 is painted pink. The background of values with an R^2 value of 0 or negative is painted brown.

Model	Optimizer	test rmse	test mae	test_R^2	Model	Optimizer	test rmse	test mae	test_R^2
LSTM(50, activation = 'relu'	Adam	339	243	0.9972	LSTM(50, activation = 'tanh'	Adam	348	248	0.997
	Nadam	429	322	0.9954		Nadam	438	327	0.9953
	RMSprop	421	289	0.9957		adamax	487	378	0.9942
	RMSprop*	500	377	0.9938		RMSprop	512	381	0.9937
	adamax	518	407	0.9935		RMSprop*	855	723	0.9814
	SGDTrue	769	586	0.9853		SGDTrue	1039	801	0.9714
	SGD	1602	1284	0.9294		SGD	1781	1437	0.9119
	adagrad	2279	1838	0.8518		adagrad	2341	1879	0.8342
	Ftrl	5526	4603	−5.2028		adadelta	4415	3535	−1.1026
	adadelta	6749	5649	−5.44		Ftrl	5196	4283	−34704
	SGDFalse	13,125	11,938	-1×10^{13}		SGDFalse	9368	8245	-6×10^{12}
LSTM(50, activation = 'elu'	Adam	345	249	0.997	LSTM(50, model.add(LeakyReLU (alpha = 0.05)) model.add(Dense(1)) model.add(LeakyReLU (alpha = 0.05))	Adam	356	258	0.9969
	adamax	444	341	0.9951		RMSprop	476	361	0.9943
	Nadam	503	394	0.9937		Nadam	480	373	0.9943
	RMSprop	515	387	0.9935		adamax	527	415	0.9932
	SGDTrue	640	473	0.9898		RMSprop*	735	612	0.9871
	SGD	1722	1386	0.9179		SGDTrue	931	706	0.9777
	adagrad	2337	1879	0.8366		SGD	1854	1502	0.8977
	RMSprop*	1,963,629	166,538	−0.0027		adagrad	2497	1982	0.7888
	adadelta	4540	3641	−1,4515		adadelta	5582	4530	−10.04
	Ftrl	5233	4319	−3,6805		Ftrl	5595	4650	−6.63
	SGDFalse	27,722	26,952	-5×10^{13}		SGDFalse	19644	18541	-2×10^{9}
LSTM(50, activation = 'sigmoid'	Adam	572	425	0.9919	LSTM(50, activation = 'softmax'	RMSprop	674	514	0.9883
	adamax	604	478	0.9909		RMSprop*	707	582	0.9869
	Nadam	651	528	0.9893		Nadam	738	551	0.9861
	RMSprop	677	555	0.9883		Adam	750	576	0.9856
	RMSprop*	735	603	0.9857		adamax	1010	804	0.9736
	SGDTrue	954	742	0.9772		SGDFalse	7976	7769	−1.0162
	SGD	1311	1040	0.9514		SGDTrue	7285	6206	−1506.18
	adagrad	4911	3931	−4.5018		adagrad	6478	5000	-6×10^{4}
	adadelta	6292	5116	−87.23		SGD	6547	5237	-2×10^{5}
	Ftrl	6314	5103	−219.58		adadelta	11096	9103	-2×10^{5}
	SGDFalse	7131	5998	-3×10^{8}		Ftrl	6484	5017	-2×10^{5}
GRU(50, activation = 'relu'	Adam	412	296	0.996	GRU(50, activation = 'tanh'	Adam	385	291	0.9964
	Nadam	405	296	0.996		RMSprop	457	339	0.995
	adamax	481	366	0.9945		Nadam	454	355	0.9948
	RMSprop	475	347	0.9944		adamax	504	393	0.9939
	RMSprop*	659	537	0.9887		SGDTrue	988	747	0.9743
	SGDTrue	962	721	0.9754		SGD	1304	1059	0.9563
	SGD	1336	1082	0.9546		SGDFalse	1354	1105	0.953
	adagrad	1935	1535	0.8824		adagrad	1772	1445	0.9156
	Ftrl	4801	3922	−1.7584		adadelta	4080	3355	−0.4021
	adadelta	6497	5391	−5.6841		RMSprop*	5444	4511	−0.5166
	SGDFalse	12,246	10,396	-4×10^{13}		Ftrl	4342	3477	−0.6307

Table 6. *Cont.*

Model	Optimizer	test rmse	test mae	test_R^2	Model	Optimizer	test rmse	test mae	test_R^2
GRU(50, activation = 'elu'	Adam	373	276	0.9965	GRU(50, model.add(LeakyReLU(alpha = 0.05)) model.add(Dense(1)) model.add(LeakyReLU(alpha = 0.05))	Adam	407	313	0.996
	Nadam	492	379	0.9938		adamax	434	330	0.9954
	adamax	535	417	0.993		Nadam	435	337	0.9954
	RMSprop	593	483	0.9914		RMSprop	485	365	0.9944
	SGDTrue	1064	815	0.9701		SGDTrue	889	664	0.9794
	SGD	1316	1070	0.9559		RMSprop*	1181	1045	0.9601
	adagrad	1809	1471	0.9117		SGD	1349	1098	0.9531
	Ftrl	4373	3503	−0.6797		adagrad	1995	1606	0.8833
	adadelta	4296	3513	−0.8821		Ftrl	4923	4032	−2.1815
	RMSprop*	38,569	38,093	−19.488		adadelta	5488	4556	−3.2035
	SGDFalse	error	error	error		SGDFalse	34,730	34,118	0
GRU(50, activation = 'sigmoid'	Adam	486	365	0.9942	GRU(50, activation = 'softmax'	Nadam	580	427	0.9914
	adamax	557	404	0.9923		Adam	582	437	0.9913
	RMSprop*	586	473	0.9909		RMSprop	605	480	0.9907
	Nadam	684	563	0.9881		adamax	760	588	0.9851
	SGDTrue	1099	875	0.9702		RMSprop*	772	590	0.984
	RMSprop	1566	1440	0.9373		SGDTrue	1467	1163	0.9472
	SGD	1729	1373	0.9118		adagrad	6507	5231	−589.92
	adagrad	5009	3992	−4.7411		SGD	6418	5136	−1050.57
	adadelta	6401	5206	−25.411		adadelta	9056	6966	−1384.89
	Ftrl	5875	4734	−28.589		Ftrl	6550	5234	-3×10^{12}
	SGDFalse	17,976	16,764	-3×10^{14}		SGDFalse	16,742	15,610	-2×10^{13}
SimpleRNN(50, activation = 'relu'	Nadam	368	266	0.9968	SimpleRNN(50, activation = 'tanh'	Nadam	444	332	0.9952
	Adam	425	307	0.9956		adamax	499	391	0.9939
	RMSprop	486	357	0.9942		Adam	504	381	0.9938
	adamax	567	441	0.9921		SGDTrue	579	430	0.9917
	SGDTrue	668	504	0.9889		RMSprop	637	511	0.9899
	RMSprop*	813	665	0.9832		SGD	988	743	0.9748
	SGD	904	685	0.9792		adagrad	1818	1385	0.911
	adagrad	1857	1488	0.9031		Ftrl	2619	2056	0.6899
	Ftrl	3174	2492	0.4742		adadelta	3752	2969	0.295
	adadelta	5072	4243	−0.5113		RMSprop*	8553	7295	−9.0302
	SGDFalse	17,577	16,467	-2×10^{12}		SGDFalse	16,187	15,041	0
SimpleRNN(50, activation = 'elu'	adamax	430	329	0.9956	SimpleRNN(50, model.add(LeakyReLU(alpha = 0.05)) model.add(Dense(1)) model.add(LeakyReLU(alpha = 0.05))	adamax	434	330	0.9955
	Nadam	454	348	0.9949		Adam	472	370	0.9946
	Adam	504	391	0.9937		Nadam	481	365	0.9944
	RMSprop	627	487	0.9902		RMSprop	496	355	0.9941
	SGDTrue	637	474	0.9896		SGDTrue	702	549	0.988
	SGD	903	684	0.9789		SGD	1147	881	0.966
	adagrad	1771	1351	0.9182		adagrad	2053	1611	0.8802
	Ftrl	2352	1839	0.7707		RMSprop*	3541	3189	0.4623
	adadelta	3982	3055	0.5751		adadelta	4410	3462	0.0894
	RMSprop*	error	error	error		Ftrl	4199	3377	−0.3491
	SGDFalse	error	error	error		SGDFalse	47,858	47,417	-2×10^{15}
SimpleRNN(50, activation = 'sigmoid'	Adam	575	430	0.9918	SimpleRNN(50, activation = 'softmax'	Adam	744	568	0.9859
	adamax	587	436	0.9913		adamax	759	595	0.9856
	Nadam	603	462	0.991		RMSprop*	802	626	0.9832
	RMSprop	767	622	0.9851		Nadam	822	637	0.9828
	SGDTrue	907	718	0.9798		RMSprop	836	649	0.9827
	RMSprop*	981	849	0.9759		SGDTrue	1469	1174	0.9484
	SGD	1407	1123	0.9477		SGD	6319	5068	−280.15
	adagrad	4848	3833	−3.4409		adagrad	6492	5215	−363.93
	Ftrl	5448	4472	−11.308		adadelta	9207	7114	−957.72
	adadelta	6623	5303	−16.200		SGDFalse	6535	5203	-1×10^{12}
	SGDFalse	8772	6677	-2×10^{13}		Ftrl	6550	5234	-2×10^{13}
Bidirectional(LSTM(50, activation = 'relu'	Adam	313	224	0.9976	Bidirectional(LSTM(50, activation = 'tanh'	Adam	449	327	0.9953
	Nadam	345	256	0.9971		adamax	469	356	0.9947
	adamax	458	342	0.9948		Nadam	486	385	0.9942
	RMSprop	564	458	0.9923		RMSprop	547	421	0.9927
	RMSprop*	656	519	0.989		RMSprop*	759	650	0.9854
	SGDTrue	832	636	0.9829		SGDTrue	1114	847	0.9678
	SGD	1481	1165	0.9419		SGD	1521	1182	0.9395
	adagrad	2212	1728	0.8649		adagrad	2135	1653	0.8748
	Ftrl	4941	4152	−2.66		adadelta	3372	2585	0.3352
	adadelta	6297	5378	−8.6355		Ftrl	4398	3642	−0.8918
	SGDFalse	11,979	10,798	0		SGDFalse	26,463	25,656	-3×10^{15}

Table 6. *Cont.*

Model	Optimizer	test rmse	test mae	test_R^2	Model	Optimizer	test rmse	test mae	test_R^2
Bidirectional(LSTM(50, activation = 'elu'	adamax	407	310	0.996	Bidirectional(LSTM(50, model.add(LeakyReLU(alpha = 0.05)) model.add(Dense(1)) model.add(LeakyReLU(alpha = 0.05))	Adam	346	255	0.9971
	Adam	409	295	0.996		Nadam	376	275	0.9966
	Nadam	457	361	0.9949		adamax	423	320	0.9956
	RMSprop	490	368	0.9941		RMSprop	569	443	0.9923
	SGDTrue	788	604	0.9844		SGDTrue	1044	781	0.9706
	adagrad	2147	1657	0.8749		SGD	1617	1272	0.9252
	SGD	1479	1148	0.9423		adagrad	2356	1826	0.8345
	adadelta	3501	2709	0.1778		RMSprop*	4743	3693	−1.185
	Ftrl	4475	3706	−1.0879		adadelta	4762	3745	−2.6703
	SGDFalse	25,001	24,144	-3×10^{14}		Ftrl	5080	4236	−3.9808
	RMSprop*	error	error	error		SGDFalse	36,371	35,825	−14357
Bidirectional(LSTM(50, activation = 'sigmoid'	Adam	533	386	0.9929	Bidirectional(LSTM(50, activation = 'softmax'	Nadam	584	422	0.9914
	adamax	551	395	0.9924		RMSprop	625	495	0.9901
	Nadam	583	475	0.9914		Adam	660	479	0.9889
	RMSprop	648	484	0.9893		RMSprop*	669	524	0.9882
	RMSprop*	718	601	0.9868		adamax	1082	839	0.97
	SGDTrue	942	742	0.9783		SGDTrue	7147	6101	−323.77
	SGD	1796	1359	0.8969		adagrad	6490	5150	−19493
	adagrad	4410	3545	−0.9066		SGD	6525	5222	−20221
	adadelta	5906	4933	−14.449		Ftrl	6525	5189	-3×10^{5}
	Ftrl	5809	4697	−25.309		adadelta	9916	7837	−77048
	SGDFalse	32,292	31,633	-7×10^{13}		SGDFalse	18,930	18,878	−7.3997

5. Results of the Study

The values in the tables above are listed from top to bottom according to their success status for each model. It can be seen that the successful models are successful in all three RMSE, MAE, and R^2 metrics. However, in a few studies, it can be seen that one of the upper and lower trials was successful in one metric and the other in another metric. Therefore, for convenience, the R^2 metric was used as a criterion both in the listing and in the graphs.

The background of values with an R^2 value of 0.99 and above is painted green, and the background of values between 0.98 and 0.99 is painted pink. The background of values with an R^2 value of 0 or negative is colored brown.

In most of the studies in the literature, Adam was used as the optimizer, and ReLU was used as the activation function. In this study, methods that can compete with Adam in terms of performance and achieve higher success in some trials have been noted. According to the study results, Nadam obtained the highest values in six groups of studies, Adamax in three groups of studies, and RMSprop in one group of studies. In other groups, the Adam method reached the highest values. In this respect, Nadam and Adamax, which are not prevalent in the literature, attract attention.

It is observed that the majority of the study results tested with the FTRL, Adadelta, and SGDfalse methods are painted with a brown background and were unsuccessful. It can be seen that the FTRL method obtained positive values only in five trials and reached an R^2 value of 0.91 in only one trial. It can be seen that the Adadelta method obtained positive values only in six studies and reached an R^2 value of 0.76 in one study. It can be seen that the SGDfalse method obtained a positive value in only one trial and reached an R^2 value of 0.95. Therefore, these three methods are not taken into account in graphical drawings. Since lower values were obtained in the experiments conducted with the Adagrad method compared with other methods, the results of this method have also been removed from some graphs. In a continuation of the fifth section, the titles in the fourth section were analyzed and the findings obtained were evaluated. Therefore, the work continues without changing the titles. However, in order to better compare successful models with each other, models with low performance in some graphs have been removed from the graph.

We have also used some methods to solve the overfitting problem in our models. Cross-validation and sliding window splitting are two of these methods. In cross-validation, the

data is divided into several subsets and the performance of the machine learning model is evaluated. Dividing the data into training and test data serves this purpose. We also tried to predict the next hour using the last 24 h of data for six columns. In fact, we created a separate dataset for each prediction and cross-validated it. We also used the sliding window method by shifting the data we use in each prediction by one hour and using the next 24 h of values.

5.1. Results of the Estimation Made Using Four Different Models and 11 Different Optimization Methods

Figures 5 and 6 are plotted for the results of the seven most successful optimization methods according to the four models. In this study, ReLU was used as the activation function, and the study was conducted over 50 epochs. According to Figure 5, when the most successful results are looked at, Adam and Nadam come first, followed by Adamax and RMSprop. The Adagrad and SGD methods are not shown in Figure 5 because of their lower performance. In Figure 6, the Adagrad, SGD, and SGDTrue methods are not shown. According to Figure 6, BiLSTM and LSTM stand out as the most successful models, although there is no clear distinction. This conclusion is supported by similar studies conducted with different datasets, which reached R^2 values of 0.99 and MAE values of 0.092 [50,51].

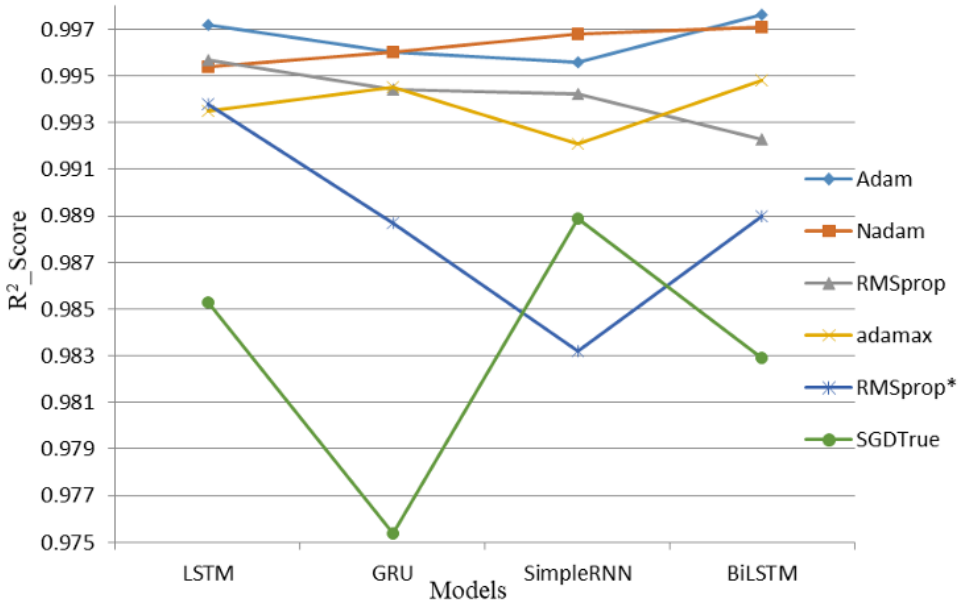


Figure 5. Four models–six optimizers–R^2 graph.

5.2. Results of the Study Using Four Different Models, 11 Different Optimizers, and 10–50–100 Epochs

In Figures 7–10 below, this study evaluated eight different optimizers using LSTM, GRU, Simple RNN, and BiLSTM models and the ReLU activation function, respectively. All studies performed for epoch values 10, 50, and 100 were evaluated. In this study, it can be seen that increasing the number of epochs has a positive effect on the results in most trials. In fact, performance decreases after a certain number of epochs for each model. However, since 100 epochs is considered sufficient for now, epoch values were not increased further. Another study observed that performance increased with the number of epochs, but that performance did not increase further after a certain number of epochs [52]. In fact, in order to save time and avoid bad results, the epoch value was taken as 50 in all studies in other sections. In Figures 8 and 9, the Adagrad method, and in Figures 7 and 10, the Adagrad and SGD methods were removed from the graph due to their poor performance.

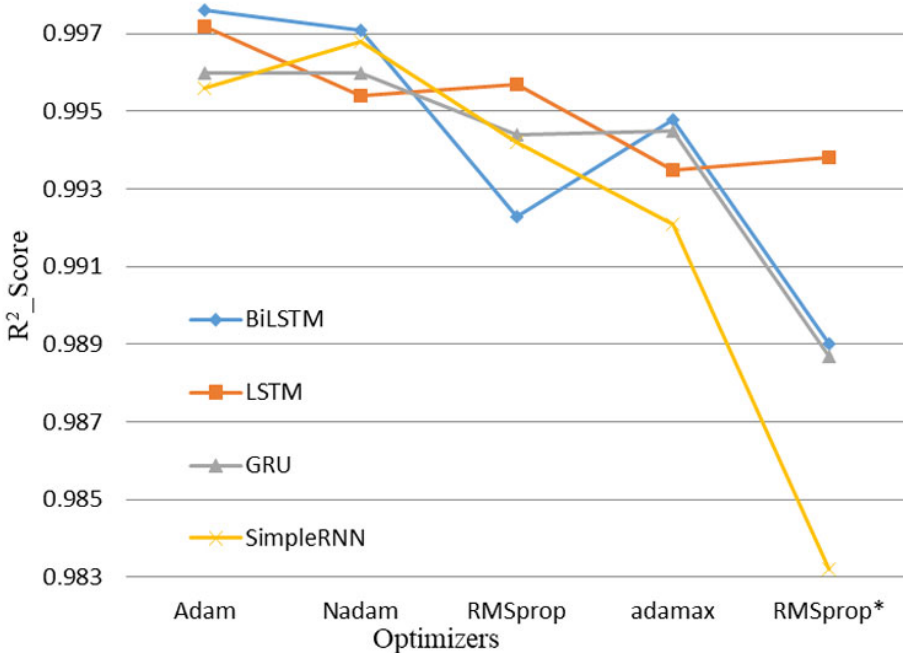


Figure 6. Four models–five optimizers–R^2 graph.

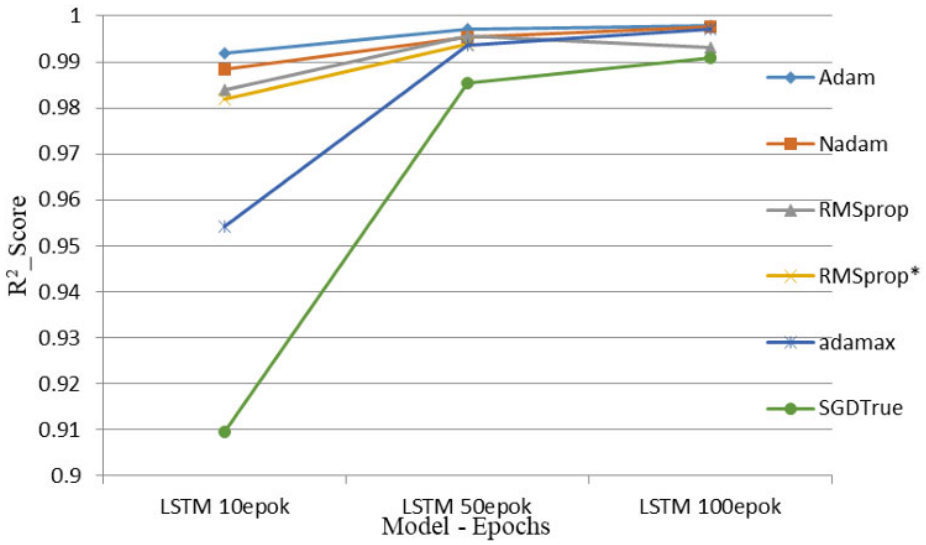


Figure 7. LSTM: six optimizers–10–50–100 epochs–R^2 graph.

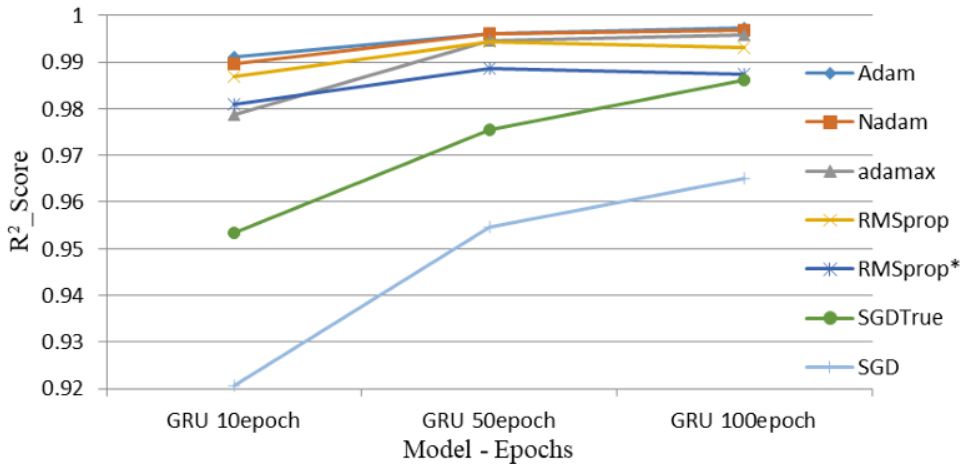


Figure 8. GRU: seven optimizers–10–50–100 epochs–R^2 graph.

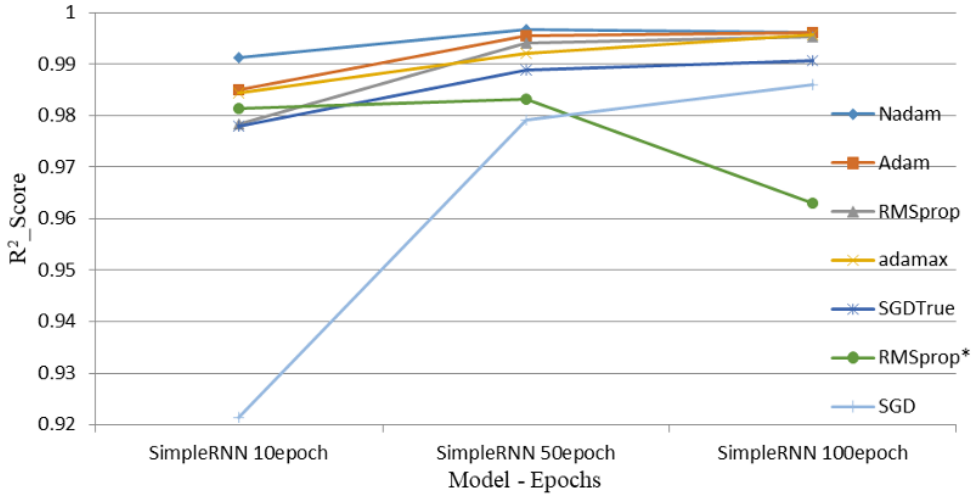


Figure 9. SimpleRNN: seven optimizers–10–50–100 epochs–R^2 graph.

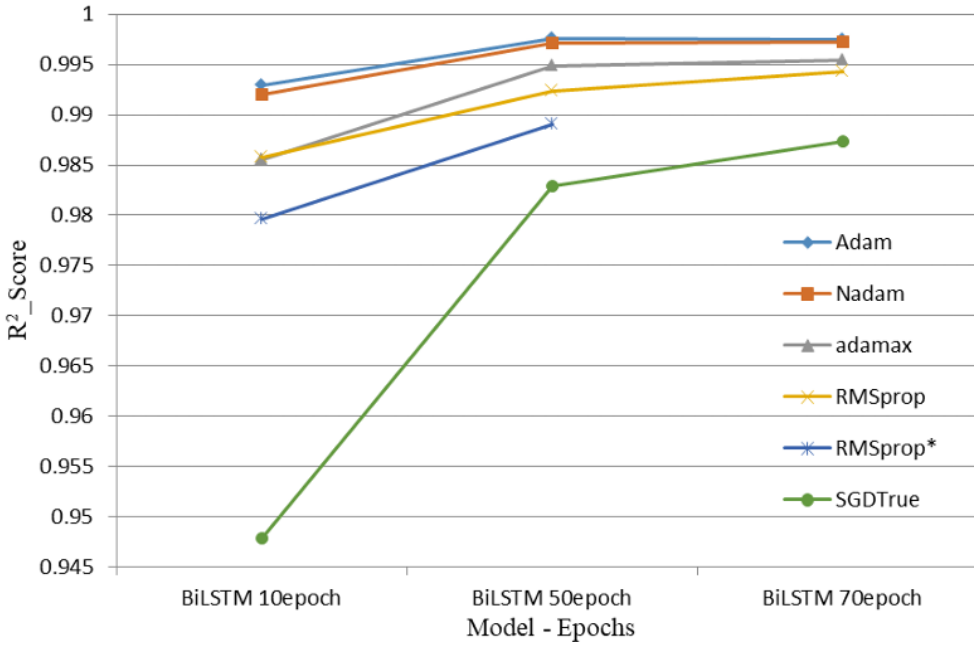


Figure 10. BiLSTM: six optimizers–10–50–70 epochs–R^2 graph.

In Figure 7, for the RMSprop* method, the epoch value of 100 was removed from the graph because it was too low. The R^2 value increases as the number of epochs increases, while for both RMSprop and RMSprop*, the value increases at 50 epochs but decreases at 100 epochs.

In Figure 8, as the number of epochs increases, the R^2 value increases, while for RMSprop this value increases at 50 epochs and decreases at 100 epochs.

In Figure 9, as the number of epochs increases, the R^2 value increases, while for both RMSprop* and Nadam, the value increases at 50 epochs but decreases at 100 epochs. Despite this, the Nadam method stands out as the most successful model.

In Figure 10, as the number of epochs increases, the R^2 value increases, while for Adam, the value increases at 50 epochs but decreases at 100 epochs. In addition, the 70 epoch value could not be measured in the RMSprop* method because it gave a NaN error due to the exploding gradients problem.

5.3. Results of the Study Using Four Different Models, Six Different Activation Functions, and 11 Different Optimizers

In Figures 11–14 below, the results of the experiments with four different architectures (LSTM, GRU, SimpleRNN, and BiLSTM), six different activation functions (ReLU, tanh, ELU, LeakyReLU, sigmoid, softmax), and six different optimizers are plotted. The epoch value is taken as 50. The Adagrad method is not shown in the graph because it obtained negative values in 8 out of 24 trials. The SGD method is not shown in the graph due to its low performance. Additionally, negative values in other methods are not shown.

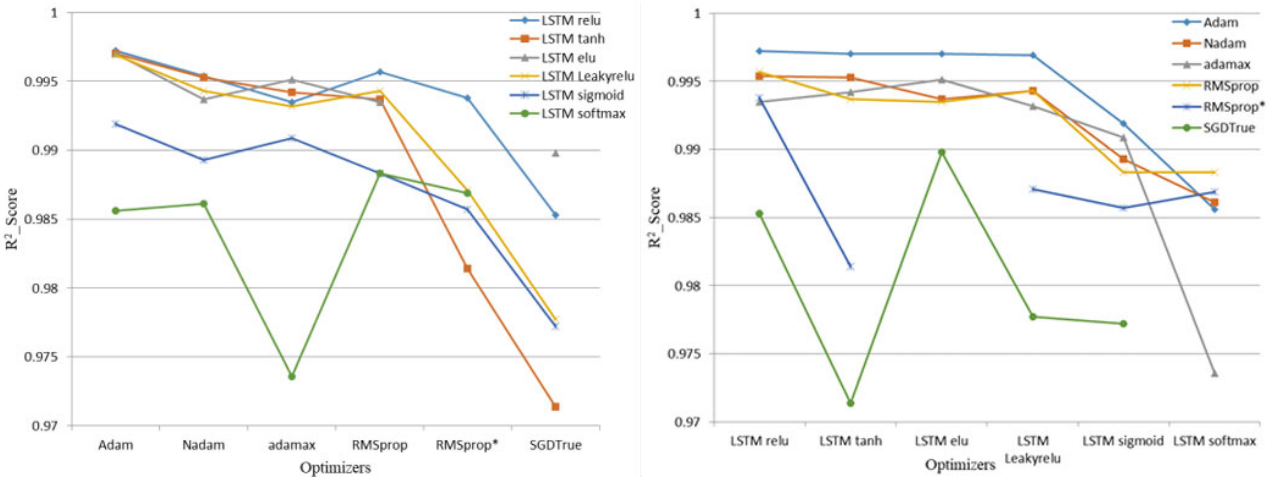


Figure 11. (**a**) Six optimizers–R^2 plots of six activation functions with LSTM, (**b**) LSTM–six activation functions–R^2 plots of six optimizers.

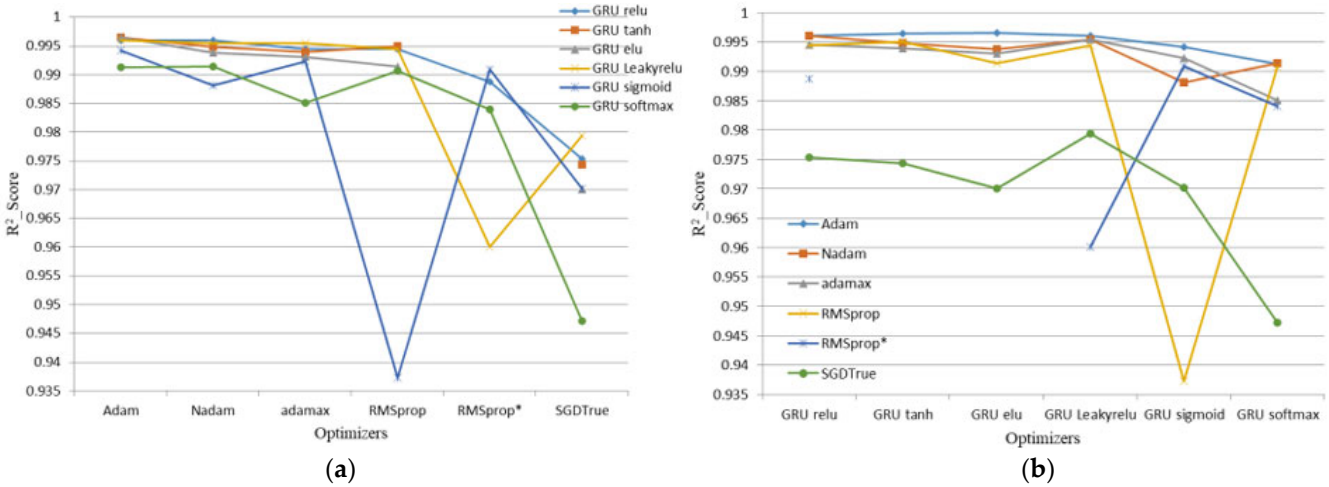
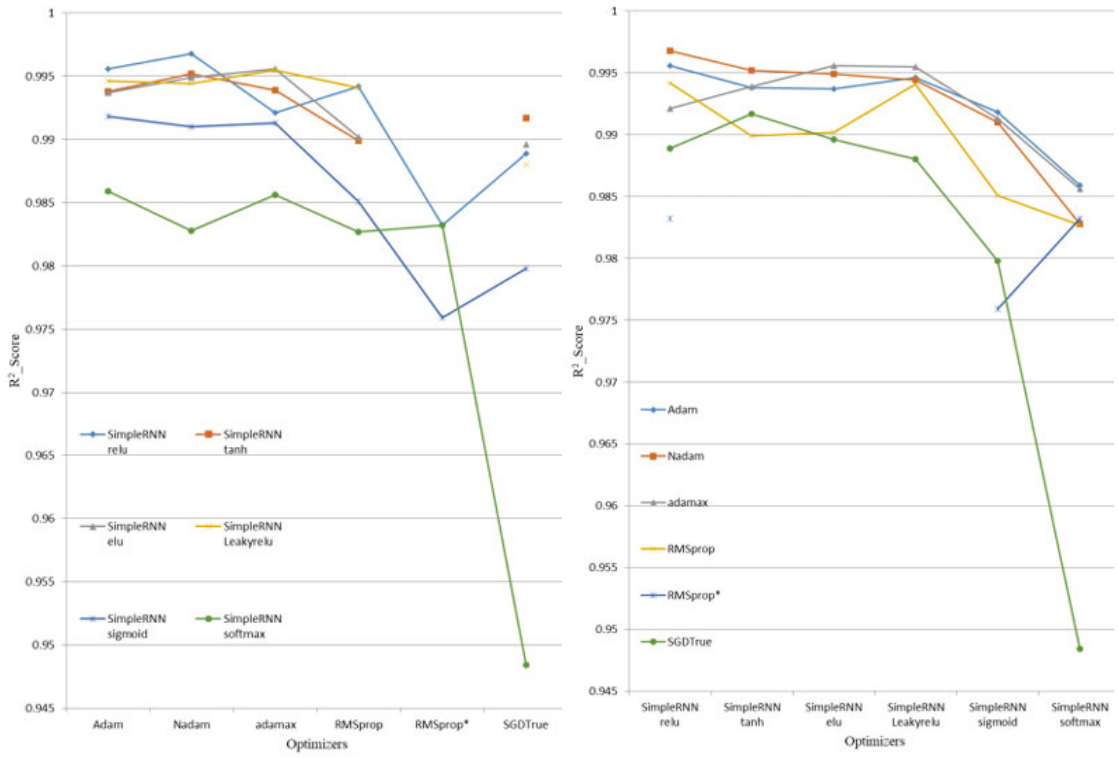


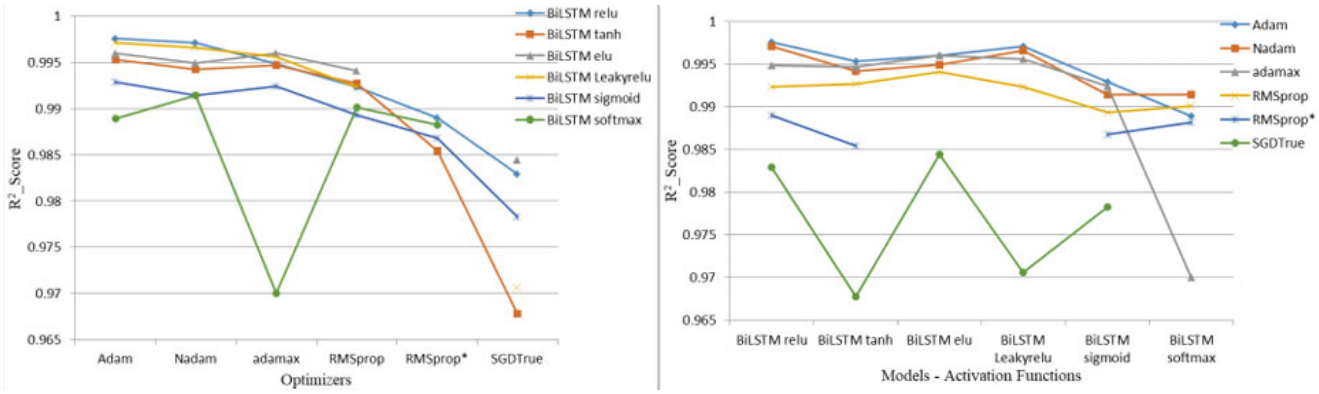
Figure 12. (**a**) Six optimizers–R^2 plots of six activation functions with GRU, (**b**) GRU–six activation functions–R^2 plots of six optimizers.

As can be seen from Figures 11a, 12a, 13a and 14a, the most successful activation functions are ReLU, tanh, ELU, and LeakyReLU. It can be seen that sigmoid and softmax are less successful. In Figure 13a,b, the study using the SimpleRNN–LeakyReLU–RMSprop* methods was deleted because its result value was low. Moreover, the SimpleRNN–ELU–RMSprop* model in Figure 13a,b and the BiLSTM–ELU–RMSprop* model in Figure 14a,b suffer from the exploding gradients problem.



(**a**) (**b**)

Figure 13. (**a**) Six optimizers–R^2 plots of six activation functions with Simple RNN, (**b**) Simple RNN–six activation functions–R^2 plots of six optimizers.



(**a**) (**b**)

Figure 14. (**a**) Six optimizers–R^2 plots of six activation functions with BiLSTM, (**b**) BiLSTM–six activation functions–R^2 plots of six optimizers.

As can be seen from Figures 11b, 12b, 13b and 14b, the most successful optimizers are Adam, Nadam, Adamax, and RMSprop. It can be seen that RMSprop* and SGDtrue are less successful.

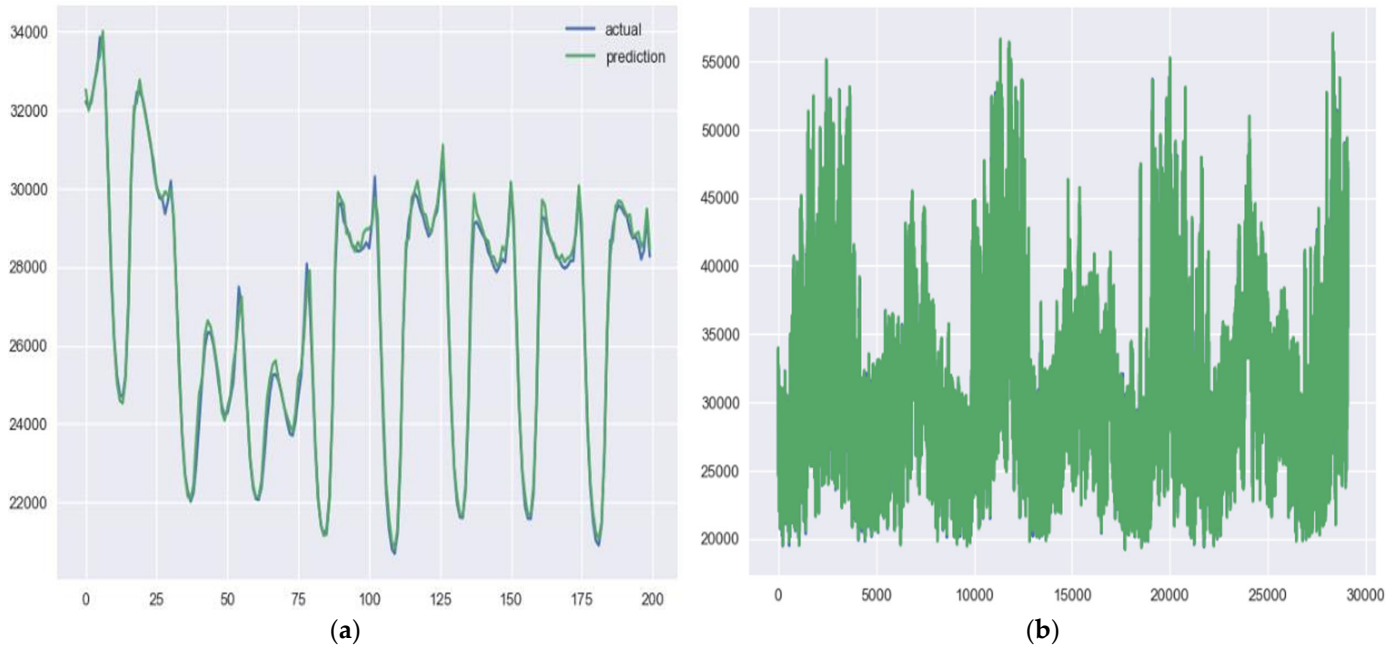
It can be seen that the SGDtrue method is more successful than the SGD method in all of the studies except one trial. When compared with these two methods, the SGDfalse method is seen to be quite unsuccessful. Additionally, throughout the study, it can be observed that the RMSprop method is more successful than the RMSprop* method, except in two trials.

5.4. Best Results Achieved

The 10 most successful studies according to MAE, RMSE, and R^2_score indices are listed in Table 7. In all these experiments, the epoch value was taken as 50. The most successful study was obtained with the BiLSTM model, the ReLU activation function, and the Adam optimization function. In the study, a value of 0.9976 was reached for R^2. Another study using the same dataset reached an R^2 value of 0.9913, which was presented as the highest value obtained [53]. Another study using the same dataset reached an R^2 value of 0.98 [54]. In a similar study, applying random hyperparameter optimisation to the LSTM model reduced the error rate to below 1.5% [52]. The graph of the real and predicted values of the first 200 and 30,000 values of this model can be seen in Figure 15a,b. Figure 16 shows how close the predictions are to the real value line. In other models, the success ranking decreases from the highest to the lowest.

Table 7. Top 10 results of studies conducted with 50 epochs. The background of values with an R^2 value of 0.99 and above is painted green.

Model	Layers	Activatör	Optimizer	Epochs	test_rmse	test-mae	test_R^2
BiLSTM	50	relu	Adam	50	313.04	223.66	0.9976
LSTM	50	relu	Adam	50	339.33	242.54	0.9972
BiLSTM	50	relu	Nadam	50	345.03	255.65	0.9971
BiLSTM	50	Leakyrelu	Adam	50	345.7	255.37	0.9971
LSTM	50	elu	Adam	50	345.19	249	0.997
LSTM	50	tanh	Adam	50	348.22	248.38	0.997
LSTM	50	Leakyrelu	Adam	50	356.31	257.81	0.9969
SimpleRNN	50	relu	Nadam	50	368.15	265.96	0.9968
BiLSTM	50	Leakyrelu	Nadam	50	375.89	275.41	0.9966
GRU	50	elu	Adam	50	373.14	275.81	0.9965



(**a**) (**b**)

Figure 15. Actual value-prediction graphs of the first 200 (**a**), and 30,000 (**b**) values according to the most successful model.

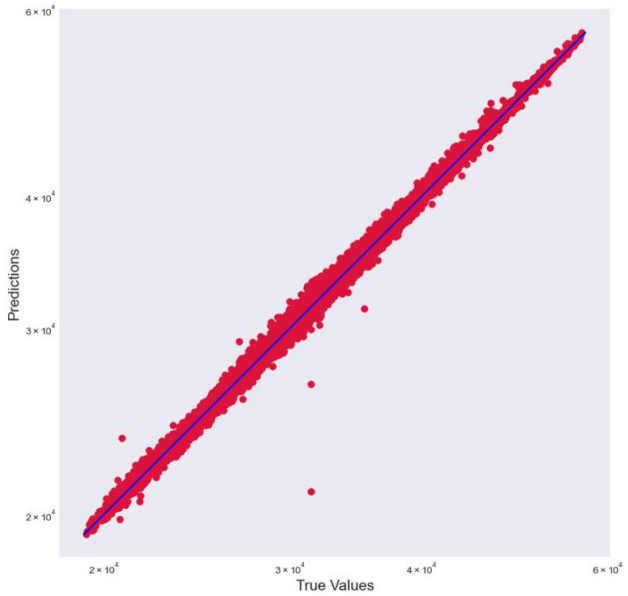


Figure 16. True value line and prediction population for all values according to the most successful model. The blue line in the figure shows the true values, and the red dots show our predictions.

The graph of the 10 most successful models according to Table 7 according to their activation function is shown in Figure 17. Accordingly, successful models can be listed as BiLSTM–Adam, LSTM–Adam, BiLSTM–Nadam, Simple RNN–Nadam, and GRU–Adam, respectively.

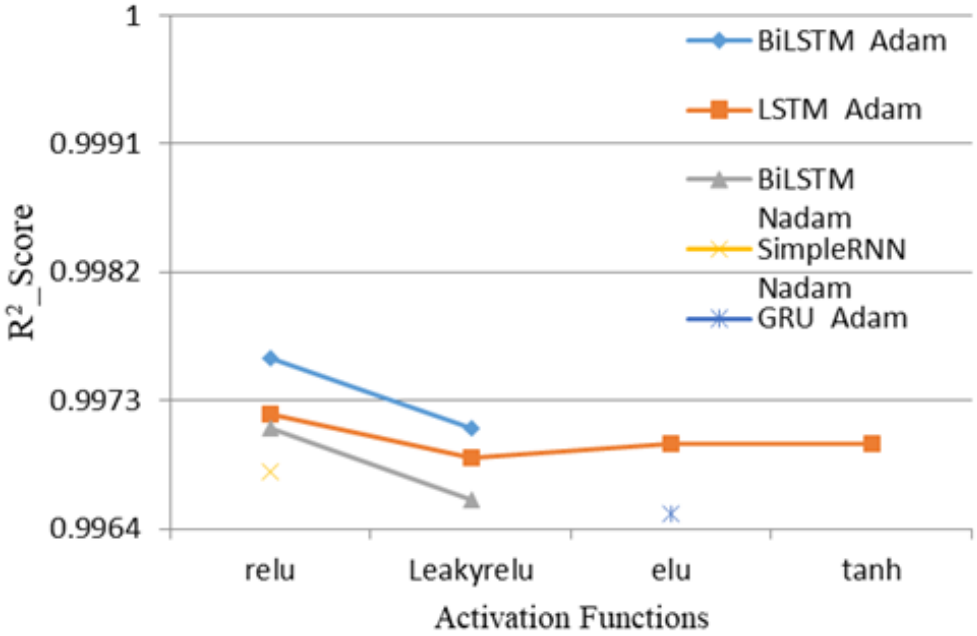


Figure 17. Activation function–R^2 plots according to the top 10 results.

The graph of the 10 most successful models according to Table 7, by model and optimizer, is shown in Figure 18. Accordingly, the most successful activation functions can be listed as ReLU, LeakyReLU, ELU, and tanh, respectively.

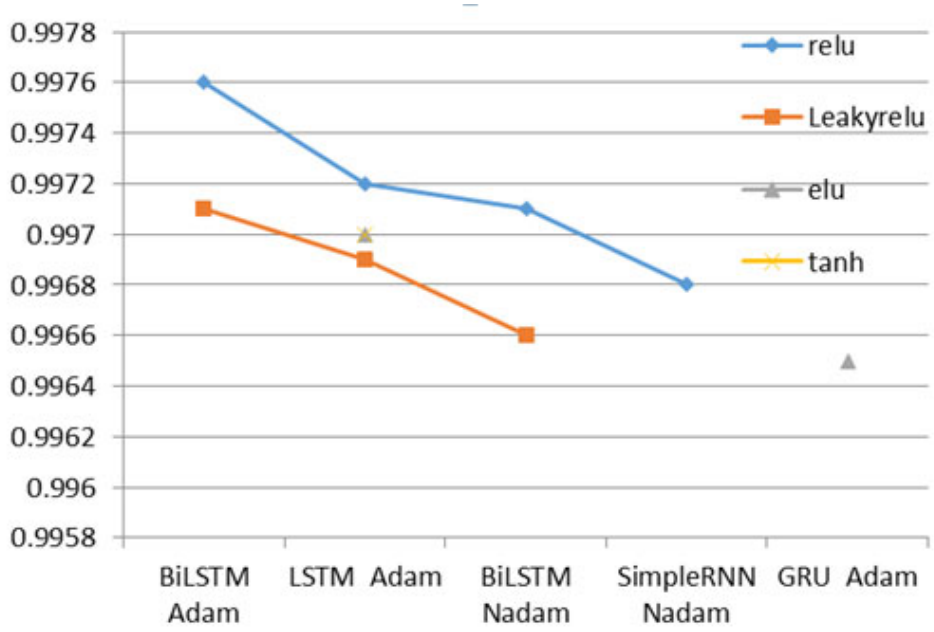


Figure 18. Model-optimizer–R^2 plots according to top 10 results.

In our study, we observed that performance varies according to the architectures and models employed. The same dataset produces different results depending on the method used. However, applying the same methods to different datasets will also produce different results. Indeed, studies have emphasized that data-driven approaches will play a significant role in estimation accuracy, and that activation function performance largely depends on the dataset [55,56].

6. Discussion, and Conclusions

The contributions of this article to the literature are listed as follows:

- Emphasizing the importance of electricity consumption estimation in smart grid energy systems.
- Showing that more accurate data will be obtained with data editing EDA.
- Showing that there are actually many successful options and models, even though single options are usually offered in similar studies.
- Showing the success of the proposed models in estimating energy consumption correctly. Also, obtaining different models and successful results with different variations.
- Outlining statistical metrics with RMSE, MAE, and R^2 to evaluate model performances.

In this study, 89 of the 352 experiments yielded R^2 values of 0.95 and above. This corresponds to a rate of 25%. In five trials, the problem of exploding gradients was encountered. Three of these occurred while working with the RMSprop* method and two while working with the SGDfalse method.

First of all, it can be seen that the BiLSTM and the LSTM models are the most successful models.

The most successful optimizers are Adam, Nadam, Adamax, and RMSprop, respectively. RMSprop* and SGDtrue seem to be less successful. It was seen that 96.5% of the studies with FTRL, Adadelta, and SGDfalse methods offered very poor results. In the literature, the Adam optimization method is generally used. Of the 32 group studies, Nadam gave the highest values in six, Adamax in three, and RMSprop in one. In this respect, Nadam and Adamax, which do not feature much in the literature, attract attention.

It can be seen that increasing the number of epochs has a positive effect on the results.

The most successful activation functions can be listed as ReLU, LeakyReLU, ELU, and tanh, respectively. Sigmoid and softmax appear to be less successful. Most of the studies in the literature were conducted with the ReLU method. When we look at the most successful models in the study, the use of 40% ReLU, 30% LeakyReLU, 20% ELU, and 10% tanh is noteworthy.

It can be seen that the SGDtrue method is more successful than the SGD method, while the SGDfalse method is quite unsuccessful. It can also be seen that the RMSprop method is more successful than the RMSprop* method.

Looking at the 10 most successful studies, the R^2 metric reaches 0.9976 with the use of BiLSTM, ReLU, and Adam.

In most studies, joint studies were carried out, such as forecasting electricity demand, production, or consumption. However, a common method or rule for model development has not been produced. Therefore, in this study, different techniques were compared, different models were applied, and model suggestions were made using different hyperparameters.

The reason for the high success in this study is that there is seasonality in our data, and the consumption data is more regular. It is proposed that the methods here will be successful when applied to similar datasets, but their performance will decrease on different datasets.

Author Contributions: Conceptualization, M.T.U. and A.K.; methodology, M.T.U. and A.K.; software, M.T.U. and A.K.; validation, M.T.U. and A.K.; formal analysis, M.T.U. and A.K.; investigation, M.T.U. and A.K.; resources, M.T.U. and A.K.; data curation, M.T.U. and A.K.; writing—original draft preparation, M.T.U. and A.K.; writing—review and editing, M.T.U. and A.K.; visualization, M.T.U. and A.K.; supervision, M.T.U. and A.K.; project administration, M.T.U. and A.K.; funding acquisition, M.T.U. and A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/yusuficolab/MDPI-Makalesi/tree/main accessed on 1 May 2025.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ali, M.; Khan, D.M.; Alshanbari, H.M.; El-Bagoury, A.A.-A.H. Prediction of Complex Stock Market Data Using an Improved Hybrid EMD-LSTM Model. *Appl. Sci.* **2023**, *13*, 1429. [CrossRef]
2. Mndawe, S.T.; Paul, B.S.; Doorsamy, W. Development of a Stock Price Prediction Framework for Intelligent Media and Technical Analysis. *Appl. Sci.* **2022**, *12*, 719. [CrossRef]
3. Jarrah, M.; Derbali, M. Predicting Saudi Stock Market Index by Using Multivariate Time Series Based on Deep Learning. *Appl. Sci.* **2023**, *13*, 8356. [CrossRef]
4. Huang, D.; Zhang, Q.; Wen, Z.; Hu, M.; Xu, W. Research on a Time Series Data Prediction Model Based on Causal Feature Weight Adjustment. *Appl. Sci.* **2023**, *13*, 10782. [CrossRef]
5. Cheng, C.-H.; Chen, Y.-S. Fundamental Analysis of Stock Trading Systems using Classification Techniques. In Proceedings of the 2007 International Conference on Machine Learning and Cybernetics, Hong Kong, China, 19–22 August 2007; pp. 1377–1382.
6. Vargas, M.R.; de Lima, B.S.L.P.; Evsukoff, A.G. Deep Learning for Stock Market Prediction from Financial News Articles. In Proceedings of the 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Annecy, France, 26–28 June 2017; pp. 60–65. [CrossRef]
7. Li, C.; Zhao, M.; Liu, Y.; Xu, F. Air Temperature Forecasting using Traditional and Deep Learning Algorithms. In Proceedings of the 7th International Conference on Information Science and Control Engineering (ICISCE), Changsha, China, 18–20 December 2020; pp. 189–194. [CrossRef]
8. Escalona-Llaguno, M.I.; Solís-Sánchez, L.O.; Castañeda-Miranda, C.L.; Olvera-Olvera, C.A.; Martinez-Blanco, M.d.R.; Guerrero-Osuna, H.A.; Castañeda-Miranda, R.; Díaz-Flórez, G.; Ornelas-Vargas, G. Comparative Analysis of Solar Radiation Forecasting Techniques in Zacatecas, Mexico. *Appl. Sci.* **2024**, *14*, 7449. [CrossRef]

9. Lyu, C.; Eftekharnejad, S. Probabilistic Solar Generation Forecasting for Rapidly Changing Weather Conditions. *IEEE Access* **2024**, *12*, 79091–79103. [CrossRef]
10. Wu, D.; Jia, Z.; Zhang, Y.; Wang, J. Predicting Temperature and Humidity in Roadway with Water Trickling Using Principal Component Analysis-Long Short-Term Memory-Genetic Algorithm Method. *Appl. Sci.* **2023**, *13*, 13343. [CrossRef]
11. Rosca, C.-M.; Carbureanu, M.; Stancu, A. Data-Driven Approaches for Predicting and Forecasting Air Quality in Urban Areas. *Appl. Sci.* **2025**, *15*, 4390. [CrossRef]
12. Swain, D.; Kumar, M.; Nour, A.; Patel, K.; Bhatt, A.; Acharya, B.; Bostani, A. Remaining Useful Life Predictor for EV Batteries Using Machine Learning. *IEEE Access* **2024**, *12*, 134418–134426. [CrossRef]
13. Wang, H.; Wang, H.; Jiang, G.; Li, J.; Wang, Y. Early fault detection of wind turbines based on operational condition clustering and optimized deep belief network modeling. *Energies* **2019**, *12*, 984. [CrossRef]
14. Zou, Y.; Sun, W.; Wang, H.; Xu, T.; Wang, B. Research on Bearing Remaining Useful Life Prediction Method Based on Double Bidirectional Long Short-Term Memory. *Appl. Sci.* **2025**, *15*, 4441. [CrossRef]
15. Wu, M.; Yue, C.; Zhang, F.; Sun, R.; Tang, J.; Hu, S.; Zhao, N.; Wang, J. State of Health Estimation and Remaining Useful Life Prediction of Lithium-Ion Batteries by Charging Feature Extraction and Ridge Regression. *Appl. Sci.* **2024**, *14*, 3153. [CrossRef]
16. Alsmadi, L.; Lei, G.; Li, L. Forecasting Day-Ahead Electricity Demand in Australia Using a CNN-LSTM Model with an Attention Mechanism. *Appl. Sci.* **2025**, *15*, 3829. [CrossRef]
17. Mukhtar, M.; Oluwasanmi, A.; Yimen, N.; Qinxiu, Z.; Ukwuoma, C.C.; Ezurike, B.; Bamisile, O. Development and Comparison of Two Novel Hybrid Neural Network Models for Hourly Solar Radiation Prediction. *Appl. Sci.* **2022**, *12*, 1435. [CrossRef]
18. Ayaz Atalan, Y.; Atalan, A. Testing the Wind Energy Data Based on Environmental Factors Predicted by Machine Learning with Analysis of Variance. *Appl. Sci.* **2025**, *15*, 241. [CrossRef]
19. Energy Forecasting. A Blog by Dr. Tao Hong. Available online: http://blog.drhongtao.com/2014/10/very-short-short-medium-long-term-load-forecasting.html (accessed on 29 May 2025).
20. Kuster, C.; Rezgui, Y.; Mourshed, M. Electrical load forecasting models: A critical systematic review. *Sustain. Cities Soc.* **2017**, *35*, 257–270. [CrossRef]
21. Tan, Z.; Zhang, J.; Wang, J.; Xu, J. Day-ahead electricity price forecasting using wavelet transform combined with ARIMA and GARCH models. *Appl. Energy* **2010**, *87*, 3606–3610. [CrossRef]
22. Esener, I.I.; Yüksel, T.; Kurban, M. Artificial Intelligence Based Hybrid Structures for Short-Term Load Forecasting Without Temperature Data. In Proceedings of the 11th International Conference on Machine Learning and Applications, Boca Raton, FL, USA, 12–15 December 2012; pp. 457–462. [CrossRef]
23. Zhang, X.M.; Grolinger, K.; Capretz, M.A.M. Forecasting Residential Energy Consumption Using Support Vector Regressions. In Proceedings of the IEEE International Conference on Machine Learning and Applications, Orlando, FL, USA, 17–20 December 2018; pp. 110–117.
24. Yang, M.; Li, W.; Zhang, H.; Wang, H. Parameters Optimization Improvement of SVM on Load Forecasting. In Proceedings of the 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 27–28 August 2016; IEEE: New York, NY, USA, 2016; pp. 257–260. [CrossRef]
25. Ding, Z.; Chen, W.; Hu, T.; Xu, X. Evolutionary double attention-based long short-term memory model for building energy prediction: Case study of a green building. *Appl. Energy* **2021**, *288*, 116660. [CrossRef]
26. Fan, C.; Sun, Y.; Zhao, Y.; Song, M.; Wang, J. Deep learning-based feature engineering methods for improved building energy prediction. *Appl. Energy* **2019**, *240*, 35–45. [CrossRef]
27. Zhu, T.; Ran, Y.; Zhou, X.; Wen, Y. A Survey of Predictive Maintenance: Systems, Purposes and Approaches. *arXiv* **2024**, arXiv:1912.07383v2. [CrossRef]
28. Fan, C.; Wang, J.; Gang, W.; Li, S. Assessment of deep recurrent neural network-based strategies for short-term building energy predictions. *Appl. Energy* **2019**, *236*, 700–710. [CrossRef]
29. Kong, W.; Dong, Z.Y.; Jia, Y.; Hill, D.J.; Xu, Y.; Zhang, Y. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Trans. Smart Grid* **2019**, *10*, 841–851. [CrossRef]
30. Muzaffar, S.; Afshari, A. Short-Term Load Forecasts Using LSTM Networks. *Energy Procedia* **2019**, *158*, 2922–2927. [CrossRef]
31. Wang, C.; Yan, Z.; Li, Q.; Zhu, Z.; Zhang, C. Energy Consumption Prediction for Drilling Pumps Based on a Long Short-Term Memory Attention Method. *Appl. Sci.* **2024**, *14*, 10750. [CrossRef]
32. Siami-Namini, S.; Tavakoli, N.; Namin, A.S. A Comparison of ARIMA and LSTM in Forecasting Time Series. In Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 1394–1401. [CrossRef]
33. IDahlan, A.; Ariateja, D.; Hamami, F.; Heryanto. The Implementation of Building Intelligent Smart Energy using LSTM Neural Network. In Proceedings of the 2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS), Bandung, Indonesia, 28–30 April 2021; pp. 1–5. [CrossRef]

34. Lin, J.; Ma, J.; Zhu, J.; Cui, Y. Short-term load forecasting based on LSTM networks considering attention mechanism. *Int. J. Electr. Power Energy Syst.* **2022**, *137*, 107818. [CrossRef]
35. Wang, J.Q.; Du, Y.; Wang, J. LSTM-based long-term energy consumption prediction with periodicity. *Energy* **2020**, *197*, 117197. [CrossRef]
36. Ungureanu, S.; Topa, V.; Cziker, A.C. Deep Learning for Short-Term Load Forecasting—Industrial Consumer Case Study. *Appl. Sci.* **2021**, *11*, 10126. [CrossRef]
37. Almalki, A.J.; Wocjan, P. Forecasting Method based upon GRU-based Deep Learning Model. In Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 16–18 December 2020; pp. 534–538. [CrossRef]
38. Karim, F.; Majumdar, S.; Darabi, H. Insights Into LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access* **2019**, *7*, 67718–67725. [CrossRef]
39. Yang, S.; Yu, X.; Zhou, Y. LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. In Proceedings of the 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), Shanghai, China, 1–4 June 2020; pp. 98–101. [CrossRef]
40. Siami-Namini, S.; Tavakoli, N.; Namin, A.S. The Performance of LSTM and BiLSTM in Forecasting Time Series. In Proceedings of the IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3285–3292. [CrossRef]
41. Xu, J.; Zeng, P. Short-term Load Forecasting by BiLSTM Model Based on Multidimensional Time-domain Feature. In Proceedings of the 4th International Conference on Neural Networks, Information and Communication Engineering (NNICE), Guangzhou, China, 19–21 January 2024; pp. 1526–1530. [CrossRef]
42. Shahid, F.; Zameer, A.; Muneeb, M. Predictions for COVID-19 with deep learning models of LSTM, GRU, and Bi-LSTM. *Chaos Solitons Fractals* **2020**, *140*, 110212. [CrossRef]
43. Şişmanoğlu, G.; Koçer, F.; Önde, M.A.; Sahingoz, O.K. Price Forecasting in Stock Exchange with Deep Learning Methods. *BEU J. Sci.* **2020**, *9*, 434–445. [CrossRef]
44. Irfan, M.; Shaf, A.; Ali, T.; Zafar, M.; Rahman, S.; Mursal, S.N.F.; AlThobiani, F.; Almas, M.A.; Attar, H.M.; Abdussamiee, N.; et al. Multi-region electricity demand prediction with ensemble deep neural networks. *PLoS ONE* **2023**, *18*, e0285456. [CrossRef] [PubMed]
45. Alıoghlı, A.A.; Yıldırım Okay, F. IoT-Based Energy Consumption Prediction Using Transformers. *Gazi Univ. J. Sci. Part A Eng. Innov.* **2024**, *11*, 304–323. [CrossRef]
46. Khan, Z.A.; Ullah, A.; Haq, I.U.; Hamdy, M.; Mauro, G.M.; Muhammad, K.; Hijji, M.; Baik, S.W. Efficient Short-Term Electricity Load Forecasting for Effective Energy Management. *Sustain. Energy Technol. Assess.* **2022**, *53 Part A*, 102337. [CrossRef]
47. Energy Consumption Dataset, Kaggle. Available online: https://www.kaggle.com/datasets/raminhuseyn/energy-consumption-dataset/data (accessed on 29 May 2025).
48. Time Series Forecasting: Exploratory Data Analysis, Kaggle. Available online: https://www.kaggle.com/code/raminhuseyn/time-series-forecasting-exploratory-data-analysis (accessed on 29 May 2025).
49. Time Series Forecasting: A Practical Guide to Exploratory Data Analysis, Towards Data Science. Available online: https://towardsdatascience.com/time-series-forecasting-a-practical-guide-to-exploratory-data-analysis-a101dc5f85b1/ (accessed on 29 May 2025).
50. Mellit, A.; Pavan, A.M.; Lughi, V. Deep learning neural networks for short-term photovoltaic power forecasting. *Renew. Energy* **2021**, *172*, 276–288. [CrossRef]
51. Alizadegan, H.; Rashidi Malki, B.; Radmehr, A.; Karimi, H.; Ilani, M.A. Comparative study of long short-term memory (LSTM), bidirectional LSTM, and traditional machine learning approaches for energy consumption prediction. *Energy Explor. Exploit.* **2024**, *43*, 281–301. [CrossRef]
52. Torres, J.F.; Martínez-Álvarez, F.; Troncoso, A. A deep LSTM network for the Spanish electricity consumption forecasting. *Neural Comput. Applic* **2022**, *34*, 10533–10545. [CrossRef]
53. Alsharekh, M.F.; Habib, S.; Dewi, D.A.; Albattah, W.; Islam, M.; Albahli, S. Improving the Efficiency of Multistep Short-Term Electricity Load Forecasting via R-CNN with ML-LSTM. *Sensors* **2022**, *22*, 6913. [CrossRef] [PubMed]
54. Ali, A.N.; Etem, T. Hourly energy consumption forecasting by LSTM and ARIMA methods. *J. Comput. Electr. Electron. Eng. Sci.* **2025**, *3*, 14–20. [CrossRef]
55. Khalil, M.; McGough, A.S.; Pourmirza, Z.; Pazhoohesh, M.; Walker, S. Machine Learning, Deep Learning and Statistical Analysis for forecasting building energy consumption—A systematic review. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105287. [CrossRef]
56. Liu, J.; Ahmad, F.A.; Samsudin, K.; Hashim, F.; Kadir, M.Z.A.A. Performance Evaluation of Activation Functions in Deep Residual Networks for Short-Term Load Forecasting. *IEEE Access* **2025**, *13*, 78618–78633. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.